(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 392 895 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 90400885.1

(51) Int. Cl.5: G11C 16/06

(22) Date of filing: 30.03.90

(30) Priority: 13.04.89 US 337566

(43) Date of publication of application:
17.10.90 Bulletin 90/42

(84) Designated Contracting States:
DE FR GB

(71) Applicant: SUNDISK CORPORATION
4401 Great America Parkway, Suite 150
Santa Clara, California 95054(US)

(72) Inventor: Harari, Eliyahou
104 Auzerais Court
Los Gatos, California 95054(US)
Inventor: Norman, Robert D.
6656 Pebblewood Court
San Jose, California 95120(US)
Inventor: Mehrotra, Sanjay
735 Berkshire Place
Milpitas, California 95035(US)

(74) Representative: Kopacz, William James
83, Avenue Foch
F-75116 Paris(FR)

(54) Flash EEprom system.

(57) A system of Flash EEprom memory chips with controlling circuits serves as non-volatile memory such as that provided by magnetic disk drives. Improvements include selective multiple sector erase, in which any combinations of Flash sectors may be erased together. Selective sectors among the selected combination may also be de-selected during the erase operation. Another improvement is the ability to remap and replace defective cells with substitute cells. The remapping is performed automatically as soon as a defective cell is detected. When the number of defects in a Flash sector becomes large, the whole sector is remapped. Yet another improvement is the use of a write cache to reduce the number of writes to the Flash EEprom memory, thereby minimizing the stress to the device from undergoing too many write/erase cycling.

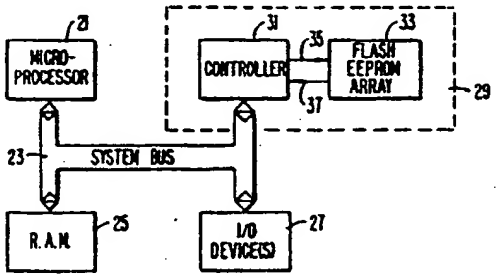


FIG._1A.

EP 0 392 895 A2

FH 008002

FLASH EEprom SYSTEM

Background of the Invention

This invention relates generally to semiconductor electrically erasable programmable read only memories (EEprom), and specifically to a system of integrated circuit Flash EEprom chips.

Computer systems typically use magnetic disk drives for mass storage of data. However, disk drives are disadvantageous in that they are bulky and in their requirement for high precision moving mechanical parts. Consequently they are not rugged and are prone to reliability problems, as well as consuming significant amounts of power. Solid state memory devices such as DRAM's and SRAM's do not suffer from these disadvantages. However, they are much more expensive, and require constant power to maintain their memory (volatile). Consequently, they are typically used as temporary storage.

EEprom's and Flash EEprom's are also solid state memory devices. Moreover, they are non-volatile, and retain their memory even after power is shut down. However, conventional Flash EEprom's have a limited lifetime in terms of the number of write (or program)/erase cycles they can endure. Typically the devices are rendered unreliable after 10^2 to 10^3 write/erase cycles. Traditionally, they are typically used in applications where semi-permanent storage of data or program is required but with a limited need for reprogramming.

Accordingly, it is an object of the present invention to provide a Flash EEprom memory system with enhanced performance and which remains reliable after enduring a large number of write/erase cycles.

It is another object of the present invention to provide an improved Flash EEprom system which can serve as non-volatile memory in a computer system.

It is another object of the present invention to provide an improved Flash EEprom system that can replace magnetic disk storage devices in computer systems.

It is another object of the present invention to provide a Flash EEprom system with improved erase operation.

It is another object of the present invention to provide a Flash EEprom system with improved error correction.

It is yet another object of the present invention to provide a Flash EEprom with improved write operation that minimizes stress to the Flash EEprom device.

It is still another object of the present invention to provide a Flash EEprom system with enhanced write operation.

Summary of the Invention

These and additional objects are accomplished by improvements in the architecture of a system of EEprom chips, and the circuits and techniques therein.

According to one aspect of the present invention, an array of Flash EEprom cells on a chip is organized into sectors such that all cells within each sector are erasable at once. A Flash EEprom memory system comprises one or more Flash EEprom chips under the control of a controller. The invention allows any combination of sectors among the chips to be selected and then erased simultaneously. This is faster and more efficient than prior art schemes where all the sectors must be erased every time or only one sector at a time can be erased. The invention further allows any combination of sectors selected for erase to be deselected and prevented from further erasing during the erase operation. This feature is important for stopping those sectors that are first to be erased correctly to the "erased" state from over erasing, thereby preventing unnecessary stress to the Flash EEprom device. The invention also allows a global de-select of all sectors in the system so that no sectors are selected for erase. This global reset can quickly put the system back to its initial state ready for selecting the next combination of sectors for erase. Another feature of the invention is that the selection is independent of the chip select signal which enables a particular chip for read or write operation. Therefore it is possible to perform an erase operation on some of the Flash EEprom chips while read and write operations may be performed on other chips not involved in the erase operation.

According to another aspect of the invention, improved error correction circuits and techniques are used to correct for errors arising from defective Flash EEprom memory cells. One feature of the invention allows defect mapping at cell level in which a defective cell is replaced by a substitute cell from the same sector. The defect pointer which connects the address of the defective cell to that of the substitute cell is stored in a defect map. Every time the defective cell is accessed, its bad data is replaced by the good data from the substitute cell.

Another feature of the invention allows defect mapping at the sector level. When the number of defective cells in a sector exceeds a predetermined number, the sector containing the defective

cells is replaced by a substitute sector.

An important feature of the invention allows defective cells or defective sectors to be remapped as soon as they are detected thereby enabling error correction codes to adequately rectify the relatively few errors that may crop up in the system.

According to yet another aspect of the present invention, a write cache is used to minimize the number of writes to the Flash EEprom memory. In this way the Flash EEprom memory will be subject to fewer stress inducing write/erase cycles, thereby retarding its aging. The most active data files are written to the cache memory instead of the Flash EEprom memory. Only when the activity levels have reduced to a predetermined level are the data files written from the cache memory to the Flash EEprom memory. Another advantage of the invention is the increase in write throughput by virtue of the faster cache memory.

According to yet another aspect of the present invention, one or more printed circuit cards are provided which contain controller and EEprom circuit chips for use in a computer system memory for long term, non-volatile storage, in place of a hard disk system, and which incorporate various of the other aspects of this invention alone and in combination.

Additional objects, features, and advantages of the present invention will be understood from the following description of its preferred embodiments, which description should be taken in conjunction with the accompanying drawings.

Brief Description of the Drawings

Fig. 1A is a general micrprocessor system including the Flash EEprom memory system of the present invention;

Fig. 1B is schematic block diagram illustrating a system including a number of Flash EEprom memory chips and a controller chip;

Fig. 2 is a schematic illustration of a system of Flash EEprom chips, among which memory sectors are selected to be erased;

Fig. 3A is a block circuit diagram in the controller for implementing selective multiple sector erase according to the preferred embodiment;

Fig. 3B shows details of a typical register used to select a sector for erase as shown in Fig. 2A;

Fig. 4 is a flow diagram illustrating the erase sequence of selective multiple sector erase;

Fig. 5 is a schematic illustration showing the partitioning of a Flash EEprom sector into a data area and a spare redundant area;

Fig. 6 is a circuit block diagram illustrating the data path control during read operation using the defect mapping scheme of the preferred embodiment;

Fig. 7 is a circuit block diagram illustrating the data path control during the write operation using the defect mapping scheme of the preferred embodiment;

Fig. 8 is a block diagram illustrating the write cache circuit inside the controller.

Description of the Preferred Embodiments EEprom System

A computer system in which the various aspects of the present invention are incorporated is illustrated generally in Figure 1A. A typical computer system architecture includes a microprocessor 21 connected to a system bus 23, along with random access, main system memory 25, and at least one or more input-output devices 27, such as a keyboard, monitor, modem, and the like. Another main computer system component that is connected to a typical computer system bus 23 is a large amount of long-term, non-volatile memory 29. Typically, such a memory is a disk drive with a capacity of tens of megabytes of data storage. This data is retrieved into the system volatile memory 25 for use in current processing, and can be easily supplemented, changed or altered.

One aspect of the present invention is the substitution of a specific type of semiconductor memory system for the disk drive but without having to sacrifice non-volatility, ease of erasing and rewriting data into the memory, speed of access, low cost and reliability. This is accomplished by employing an array of electrically erasable programmable read only memories (EEprom's) integrated circuit chips. This type of memory has additional advantages of requiring less power to operate, and of being lighter in weight than a hard disk drive magnetic media memory, thereby being especially suited for battery operated portable computers.

The bulk storage memory 29 is constructed of a memory controller 31, connected to the computer system bus 23, and an array 33 of EEprom integrated circuit chips. Data and instructions are communicated from the controller 31 to the EEprom array 33 primarily over a serial data line 35. Similarly, data and status signals are communicated from the EEprom 33 to the controller 31 over serial data lines 37. Other control and status circuits between the controller 31 and the EEprom array 33 are not shown in Figure 1A.

Referring to Figure 1B, the controller 31 is preferably formed primarily on a single integrated circuit chip. It is connected to the system address and data bus 39, part of the system bus 33, as well

as being connected to system control lines 41, which include interrupt, read, write and other usual computer system control lines.

The EEprom array 33 includes a number of EEprom integrated circuit chips 43, 45, 47, etc. Each includes a respective chip select and enable line 49, 51 and 53 from interface circuits 40. The interface circuits 40 also act to interface between the serial data lines 35, 37 and a circuit 55. Memory location addresses and data being written into or read from the EEprom chips 43, 45, 47, etc. are communicated from a bus 55, through logic and register circuits 57 and thence by another bus 59 to each of the memory chips 43, 45, 47 etc.

The bulk storage memory 29 of Figures 1A and 1B can be implemented on a single printed circuit card for moderate memory sizes. The various lines of the system buses 39 and 41 of Figure 1b are terminated in connecting pins of such a card for connection with the rest of the computer system through a connector. Also connected to the card and its components are various standard power supply voltages (not shown).

For large amounts of memory, that which is conveniently provided by a single array 33 may not be enough. In such a case, additional EEprom arrays can be connected to the serial data lines 35 and 37 of the controller chip 31, as indicated in Figure 1B. This is preferably all done on a single printed circuit card but if space is not sufficient to do this, then one or more EEprom arrays may be implemented on a second printed circuit card that is physically mounted onto the first and connected to a common controller chip 31.

Erase of Memory Structures

In system designs that store data in files or blocks the data will need to be periodically updated with revised or new information. It may also be desirable to overwrite some no longer needed information, in order to accommodate additional information. In a Flash EEprom memory, the memory cells must first be erased before information is placed in them. That is, a write (or program) operation is always preceded by an erase operation.

In conventional Flash erase memory devices, the erase operation is done in one of several ways. For example, in some devices such as the Intel corporation's model 27F-256 CMOS Flash EEprom, the entire chip is erased at one time. If not all the information in the chip is to be erased, the information must first be temporarily saved, and is usually written into another memory (typically RAM). The information is then restored into the nonvolatile Flash erase memory by programming back into the device. This is very slow and requires extra memory as holding space.

In other devices such as Seeq Technology Incorporated's model 48512 Flash EEprom chip, the memory is divided into blocks (or sectors) that are each separately erasable, but only one at a time. By selecting the desired sector and going through the erase sequence the designated area is erased. While, the need for temporary memory is reduced, erase in various areas of the memory still requires a time consuming sequential approach.

In the present invention, the Flash EEprom memory is divided into sectors where all cells within each sector are erasable together. Each sector can be addressed separately and selected for erase. One important feature is the ability to select any combination of sectors for erase together. This will allow for a much faster system erase than by doing each one independently as in prior art.

Figure 2 illustrates schematically selected multiple sectors for erase. A Flash EEprom system includes one or more Flash EEprom chips such as 201, 203, 205. They are in communication with a controller 31 through lines 209. Typically, the controller 31 is itself in communication with a microprocessor system (not shown). The memory in each Flash EEprom chip is partitioned into sectors where all memory cells within a sector are erasable together. For example, each sector may have 512 byte (i.e. 512x8 cells) available to the user, and a chip may have 1024 sectors. Each sector is individually addressable, and may be selected, such as sectors 211, 213, 215, 217 in a multiple sector erase. As illustrated in figure 2, the selected sectors may be confined to one EEprom chip or be distributed among several chips in a system. The sectors that were selected will all be erased together. This capability will allow the memory and system of the present invention to operate much faster than the prior art architectures.

Figure 3A illustrates a block diagram circuit 220 on a Flash EEprom chip (such as the chip 201 of figure 2) with which one or more sectors such as 211, 213 are selected (or deselected) for erase. Essentially, each sector such as 211, 213 is selected or tagged by setting the state of an erase enable register such as 221, 223 associated with the respective sectors. The selection and subsequent erase operations are performed under the control of the controller 31 (see figure 2). The circuit 220 is in communication with the controller 31 through lines 209. Command information from the controller is captured in the circuit 220 by a command register 225 through a serial interface 227. It is then decoded by a command decoder 229 which outputs various control signals. Similarly, address information is captured by an address register 231 and is decoded by an address decoder 233.

FH 008005

For example, in order to select the sector 211 for erase, the controller sends the address of the sector 211 to the circuit 220. The address is decoded in line 235 and is used in combination with a set erase enable signal in bus 237 to set an output 239 of the register 221 to HIGH. This enables the sector 211 in a subsequent erase operation. Similarly, if the sector 213 is also desired to be erased, its associated register 223 may be set HIGH.

Figure 3B shows the structure of the register such as 221, 223 in more detail. The erase enable register 221 is a SET/RESET latch. Its set input 241 is obtained from the set erase enable signal in bus 237 gated by the address decode in line 235. Similarly, the reset input 243 is obtained from the clear erase enable signal in bus 237 gated by the address decode in line 235. In this way, when the set erase enable signal or the clear erase enable signal is issued to all the sectors, the signal is effective only on the sector that is being addressed.

After all sectors intended for erase have been selected, the controller then issues to the circuit 220, as well as all other chips in the system a global erase command in line 251 along with the high voltage for erasing in line 209. The device will then erase all the sectors that have been selected (i.e. the sectors 211 and 213) at one time. In addition to erasing the desired sectors within a chip, the architecture of the present system permits selection of sectors across various chips for simultaneous erase.

Figures 4(1)-4(11) illustrate the algorithm used in conjunction with the circuit 220 of figure 3A. In figure 4(1), the controller will shift the address into the circuit 220 which is decoded in the line to the erase enable register associated with the sector that is to be erased. In figure 4(2), the controller shifts in a command that is decoded to a set erase enable command which is used to latch the address decode signal onto the erase enable register for the addressed sector. This tags the sector for subsequent erase. In figure 4(3), if more sectors are to be tagged, the operations described relative to figures 4(1)-4(2) are repeated until all sectors intended for erase have been tagged. After all sectors intended for erase have been tagged, the controller initiates an erase cycle as illustrated in figure 4(4).

Optimized erase implementations have been disclosed in two copending U.S. patent applications. They are copending U.S. patent applications, Serial No. 204,175, filed June 8, 1988, by Dr. Eliyahou Harari and one entitled "Multi-State EEprom Read and Write Circuits and Techniques," filed on the same day as the present application, by Sanjay Mehrotra and Dr. Eliyahou Harari. The disclosures of the two applications are hereby incorporate by reference. The Flash EEprom cells are erased by applying a pulse of erasing voltage followed by a read to verify if the cells are erased to the "erased" state. If not, further pulsing and verifying are repeated until the cells are verified to be erased. By erasing in this controlled manner, the cells are not subject to over-erasure which tends to age the EEprom device prematurely as well as make the cells harder to program.

As the group of selected sectors is going through the erase cycle, some sectors will reach the "erase" state earlier than others. Another important feature of the present invention is the ability to remove those sectors that have been verified to be erased from the group of selected sectors, thereby preventing them from over-erasing.

Returning to figure 4(4), after all sectors intended for erase have been tagged, the controller initiates an erase cycle to erase the group of tagged sectors. In figure 4(5), the controller shifts in a global command called Enable Erase into each Flash EEprom chip that is to perform an erase. This is followed in figure 4(5) by the controller raising of the erase voltage line (Ve) to a specified value for a specified duration. The controller will lower this voltage at the end of the erase duration time. In figure 4(6), the controller will then do a read verify sequence on the sectors selected for erase. In figure 4(7), if none of the sectors are verified, the sequences illustrated in figures 4(5)-4-(7) are repeated. In figures 4(8) and 3(9), if one or more sectors are verified to be erased, they are taken out of the sequence. Referring also to figure 3A, this is achieved by having the controller address each of the verified sectors and clear the associated erase enable registers back to a LOW with a clear enable command in bus 237. The sequences illustrated in figures 4(5)-4(10) are repeated until all the sectors in the group are verified to be erased in figure 4(11). At the completion of the erase cycle, the controller will shift in a No Operation (NOP) command and the global Enable Erase command will be withdrawn as a protection against a false erasure.

The ability to select which sectors to erase and which ones not to, as well as which ones to stop erasing is advantageous. It will allow sectors that have erased before the slower erased sectors to be removed from the erase sequence so no further stress on the device will occur. This will increase the reliability of the system. Additional advantage is that if a sector is bad or is not used for some reason, that sector can be skipped over with no erase occurring within that sector. For example, if a sector is defective and have shorts in it, it may consume much power. A significant system advantage is gained by the present invention which allows it to be skipped on erase cycles so that it may

greatly reduce the power required to erase the chip.

Another consideration in having the ability to pick the sectors to be erased within a device is the power savings to the system. The flexibility in erase configuration of the present invention enables the adaptation of the erase needs to the power capability of the system. This can be done by configuring the systems to be erased differently by software on a fixed basis between different systems. It also will allow the controller to adaptively change the amount of erasing being done by monitoring the voltage level in a system, such as a laptop computer.

An additional performance capability of the system in the present invention is the ability to issue a reset command to a Flash EEprom chip which will clear all erase enable latches and will prevent any further erase cycles from occurring. This is illustrated in figures 2A and 2B by the reset signal in the line 261. By doing this in a global way to all the chips, less time will be taken to reset all the erase enable registers.

An additional performance capability is to have the ability to do erase operations without regard to chip select. Once an erase is started in some of the memory chips, the controller in the system can access other memory chips and do read and write operations on them. In addition, the device(s) doing the erase can be selected and have an address loaded for the next command following the erase.

Defect Mapping

Physical defects in memory devices give rise to hard errors. Data becomes corrupted whenever it is stored in the defective cells. In conventional memory devices such as RAM's, and Disks, any physical defects arising from the manufacturing process are corrected at the factory. In RAM's spare redundant memory cells on chip may be patched on, in place of the defective cells. In the traditional disk drive, the medium is imperfect and susceptible to defects. To overcome this problem manufacturers have devised various methods of operating with these defects present, the most usual being defect mapping of sectors. In a normal disk system the media is divided into cylinders and sectors. The sector being the basic unit in which data is stored. When a system is partitioned into the various sectors the sectors containing the defects are identified and are marked as bad and not to be used by the system. This is done in several ways. A defect map table is stored on a particular portion of the disk to be used by the interfacing controller. In addition, the bad sectors are marked as bad by special ID and flag markers. When the defect is addressed, the data that would normally be stored there is placed in an alternative location. The requirement for alternative sectors makes the system assign spare sectors at some specific interval or location. This reduces the amount of memory capacity and is a performance issue in how the alternative sectors are located.

One important application of the present invention is to replace a conventional disk storage device with a system incorporating an array of Flash EEprom memory chips. The EEprom system is preferably set up to emulate a conventional disk, and may be regarded as a "solid-state disk".

In a "disk" system made from such solid-state memory devices, low cost considerations necessitate efficient handling of defects. Another important feature of the invention enables the error correction scheme to conserve as much memory as possible. Essentially, it calls for the defective cells to be remapped cell by cell rather than by throwing away the whole sector (512 bytes typically) whenever a defect occurs in it. This scheme is especially suited to the Flash EEprom medium since the majority of errors will be bit errors rather than a long stream of adjacent defects as is typical in traditional disk medium.

In both cases of the prior art RAM and magnetic disk, once the device is shipped from the factory, there is little or no provision for replacing hard errors resulting from physical defects that appear later during normal operation. Error corrections then mainly rely on schemes using error correction codes (ECC).

The nature of the Flash EEprom device predicates a higher rate of cell failure especially with increasing program/erase cycling. The hard errors that accumulate with use would eventually overwhelm the ECC and render the device unusable. One important feature of the present invention is the ability for the system to correct for hard errors whenever they occur. Defective cells are detected by their failure to program or erase correctly. Also during read operation, defective cells are detected and located by the ECC. As soon as a defective cell is identified, the controller will apply defect mapping to replace the defective cell with a space cell located usually within the same sector. This dynamic correction of hard errors, in addition to conventional error correction schemes, significantly prolongs the life of the device.

Another feature of the present invention is an adaptive approach to error correction. Error correction code (ECC) is employed at all times to correct for soft errors as well as any hard errors that may arise. As soon as a hard error is detected, defect mapping is used to replace the defective cell with a spare cell in the same sector block. Only when the number of defective cells in a sector exceeds the

defect mapping's capacity for that specific sector will the whole sector be replaced as in a conventional disk system. This scheme minimized wastage without compromising reliability.

Figure 5 illustrates the memory architecture for the cell remapping scheme. As described before, the Flash EEprom memory is organized into sectors where the cells in each sector are erasable together. The memory architecture has a typical sector 401 organized into a data portion 403 and a spare (or shadow) portion 405. The data portion 403 is memory space available to the user. The spare portion 405 is further organized into an alternative defects data area 407, a defect map area 409, a header area 411 and an ECC and others area 413. These areas contain information that could be used by the controller to handle the defects and other overhead information such as headers and ECC.

Whenever a defective cell is detected in the sector, a good cell in the alternative defects data area 407 is assigned to backup the data designated for the defective cell. Thus even if the defective cell stores the data incorrectly, an error-free copy is stored in the backup cell. The addresses of the defective cell and the backup cell are stored as defect pointers in the defect map 409.

It is to be understood that the partitioning between the user data portion 403 and the spare portion 405 need not be rigid. The relative size of the various partitioned areas may be logically reassigned. Also the grouping of the various areas is largely for the purpose of discussion and not necessarily physically so. For example, the alternative defects data area 407 has been schematically grouped under the spare portion 405 to express the point that the space it occupies is no longer available to the user.

In a read operation, the controller first reads the header, the defect map and the alternative defects data. It then reads the actual data. It keeps track of defective cells and the location of the substitute data by means of the defect map. Whenever a defective cell is encountered, the controller substitutes its bad data with the good data from the alternative defects.

Figure 6 illustrates the read data path control in the preferred embodiment. A memory device 33 which may include a plurality of Flash EEprom chips is under the control of the controller 31. The controller 31 is itself part of a microcomputer system under the control of a microprocessor (not shown). To initiate the reading of a sector, the microprocessor loads a memory address generator 503 in the controller with a memory address for starting the read operation. This information is loaded through a microprocessor interface port 505. Then the microprocessor loads a DMA controller 507 with the starting location in buffer memory or bus address that the data read should be sent. Then the microprocessor loads the header information (Head, Cylinder and sector) into a holding register file 509. Finally, the microprocessor loads a command sequencer 511 with a read command before passing control to the controller 31.

After assuming control, the controller 31 first addresses the header of the sector and verifies that the memory is accessed at the address that the user had specified. This is achieved by the following sequence. The controller selects a memory chip (chip select) among the memory device 33 and shifts the address for the header area from the address generator 503 out to the selected memory chip in the memory device 33. The controller then switches the multiplexer 513 and shifts also the read command out to the memory device 33. Then the memory device reads the address sent it and begins sending serial data from the addressed sector back to the controller. A receiver 515 in the controller receives this data and puts it in parallel format. In one embodiment, once a byte (8 bits) is compiled, the controller compares the received data against the header data previously stored by the microprocessor in the holding register file 509. If the compare is correct, the proper location is verified and the sequence continues.

Next the controller 31 reads the defect pointers and loads these bad address locations into the holding register file 509. This is followed by the controller reading the alternative defects data that were written to replace the bad bits as they were written. The alternative bits are stored in an alternative defects data file 517 that will be accessed as the data bits are read.

Once the Header has been determined to be a match and the defect pointers and alternative bits have been loaded, the controller begins to shift out the address of the lowest address of the desired sector to be read. The data from the sector in the memory device 33 is then shifted into the controller chip 31. The receiver 515 converts the data to a parallel format and transfers each byte into a temporary holding FIFO 519 to be shipped out of the controller.

A pipeline architecture is employed to provide efficient throughput as the data is gated through the controller from the receiver 515 to the FIFO 519. As each data bit is received from memory the controller is comparing the address of the data being sent (stored in the address generator 507) against the defect pointer map (stored in the register file 509). If the address is determined to be a bad location, by a match at the output of the comparator 521, the bad bit from the memory received by the receiver 515 is replaced by the

good bit for that location. The good bit is obtained from the alternative defects data file 517. This is done by switching the multiplexer 523 to receive the good bit from the alternative defects data file instead of the bad bit from the receiver 515, as the data is sent to the FIFO 519. Once the corrected data is in the FIFO it is ready to be sent to buffer memory or system memory (not shown). The data is sent from the controller's FIFO 519 to the system memory by the controller's DMA controller 507. This controller 507 then requests and gets access to the system bus and puts out an address and gates the data via an output interface 525 out to the system bus. This is done as each byte gets loaded into the FIFO 519. As the corrected data is loaded into the FIFO it will also be gated into an ECC hardware 527 where the data file will be acted on by the ECC.

Thus in the manner described, the data read from the memory device 33 is gated through the controller 31 to be sent to the system. This process continues until the last bit of addressed data has been transferred.

In spite of defect mapping of previously detected defective cells, new hard errors might occur since the last mapping. As the dynamic defect mapping constantly "puts away" new defective cells, the latest hard error that may arise between defect mapping would be adequately handled by the ECC. As the data is gated through the controller 31, the controller is gating the ECC bits into the ECC hardware 527 to determine if the stored value matched the just calculated remainder value. If it matches then the data transferred out to the system memory was good and the read operation was completed. However, if the ECC registers an error then a correction calculation on the data sent to system memory is performed and the corrected data retransmitted. The method for calculating the error can be done in hardware or software by conventional methods. The ECC is also able to calculate and locate the defective cell causing the error. This may be used by the controller 31 to update the defect map associated with the sector in which the defective cell is detected. In this manner, hard errors are constantly removed from the Flash EEprom system.

Figure 7 illustrates the write data path control in the preferred embodiment. The first portion of a write sequence is similar to a read sequence described previously. The microprocessor first loads the Address pointers for the memory device 33 and the DMA as in the read sequence. It also loads the header desired into the address generator 503 and the command queue into the command sequencer 511. The command queue is loaded with a read header command first. Thereafter, control is passed over to the controller 31. The controller then gates the address and command to the memory device 33, as in the read sequence. The memory device returns header data through controller's receiver 515. The controller compares the received header data to the expected value (stored in the holding register file 509). If the compare is correct, the proper location is verified and the sequence continues. Then the controller loads the defective address pointers from the memory device 33 into the holding register file 509 and the alternative data into the alternative defects data file 517.

Next, the controller begins to fetch the write data from system memory (not shown). It does this by getting access to the system bus, outputs the memory or bus address and does the read cycle. It pulls the data into a FIFO 601 through an input interface 603. The controller then shifts the starting sector address (lowest byte address) from the address generator 503 to the selected memory device 33. This is followed by data from the FIFO 601. These data are routed through multiplexers 605 and 513 and converted to serial format before being sent to the memory device 33. This sequence continues until all bytes for a write cycle have been loaded into the selected memory.

A pipeline architecture is employed to provide efficient throughput as the data is gated from the FIFO 601 to the selected memory 33. The data gated out of the FIFO 601 is sent to the ECC hardware 527 where a remainder value will be calculated within the ECC. In the next stage, as the data is being sent to the memory device through multiplexers 605 and 513, the comparator 521 is comparing its address from the address generator 503 to the defect pointer address values in the holding register file 509. When a match occurs, indicating that a defective location is about to be written, the controller saves this bit into the alternative defect data file 517. At the same time, all bad bits sent to memory will be sent as zeroes.

After the bytes for a write cycle have been loaded into the selected memory device, the controller issues a program command to the memory device and initiate a write cycle. Optimized implementations of write operation for Flash EEprom device have been disclosed in two previously cited co-pending U.S. patent applications, Serial No. 204,175, and one entitled "Multi-State EEprom Read and Write Circuits and Techniques." Relevant portions of the disclosures are hereby incorporated by reference. Briefly, during the write cycle, the controller applies a pulse of programming (or writing) voltages. This is followed by a verify read to determine if all the bits have been programmed properly. If the bits did not verify, the controller repeats the program/verify cycle until all bits are correctly programmed.

If a bit fails to verify after prolonged

program/verify cycling, the controller will designate that bit as defective and update the defect map accordingly. The updating is done dynamically, as soon as the defective cell is detected. Similar actions are taken in the case of failure in erase verify.

After all the bits have been programmed and verified, the controller loads the next data bits from the FIFO 601 and addresses the next location in the addressed sector. It then performs another program/verify sequence on the next set of bytes. The sequence continues until the end of the data for that sector. Once this has occurred, the controller addresses the shadow memory (header area) associated with the sector (see figure 5) and writes the contents of the ECC registers into this area.

In addition, the collection of bits that was flagged as defective and were saved in the alternative defects data file 516 is then written in memory at the alternative defects data locations (see figure 5), thereby saving the good bit values to be used on a subsequent read. Once these data groups are written and verified, the sector write is considered completed.

The present invention also has provision for defect mapping of the whole sector, but only after the number of defective cells in the sector has exceeded the cell defect mapping's capacity for that specific sector. A count is kept of the number of defective cells in each sector. When the number in a sector exceeds a predetermined value, the controller marks that sector as defective and maps it to another sector. The defect pointer for the linked sectors may be stored in a sector defect map. The sector defect map may be located in the orginal defective sector if its spare area is sufficiently defect-free. However, when the data area of the sector has accumulated a large number of defects, it is quite likely that the spare area will also be full of defects.

Thus, it is preferable in another embodiment to locate the sector map in another memory maintained by the controller. The memory may be located in the controller hardware or be part of the Flash EEprom memory. When the controller is given an address to access data, the controller compares this address against the sector defect map. If a match occurs then access to the defective sector is denied and the substitute address present in the defect map is entered, and the corresponding substitute sector is accessed instead.

In yet another embodiment, the sector remapping is performed by the microprocessor. The microprocessor looks at the incoming address and compares it against the sector defect map. If a match occurs, it does not issue the command to the controller but instead substitute the alternative location as the new command.

Apart from the much higher speed of the solid-state disk, another advantage is the lack of mechanical parts. The long seek times, rotational latency inherent in disk drives are not present. In addition, the long synchronization times, sync mark detects and write gaps are not required. Thus the overhead needed for accessing the location where data is to be read or written is much less. All of these simplifications and lack of constraints result in a much faster system with much reduced overheads. In addition, the files can be arranged in memory in any address order desired, only requiring the controller to know how to get at the data as needed.

Another feature of the invention is that defect mapping is implemented without the need to interrupt the data stream transferred to or from the sector. The data in a block which may contain errors are transferred regardless, and is corrected afterwards. Preserving the sequential addressing will result in higher speed by itself. Further, it allows the implementation of an efficient pipeline architecture in the read and write data paths.

Write Cache System

Cache memory is generally used to speed up the performance of systems having slower access devices. For example in a computer system, access of data from disk storage is slow and the speed would be greatly improved if the data could be obtained from the much faster RAM. Typically a part of system RAM is used as a cache for temporarily holding the most recently accessed data from disk. The next time the data is needed, it may be obtained from the fast cache instead of the slow disk. The scheme works well in situations where the same data is repeatedly operated on. This is the case in most structures and programs since the computer tends to work within a small area of memory at a time in running a program. Another example of caching is the using of faster SRAM cache to speed up access of data normally stored in cheaper but slower DRAM.

Most of the conventional cache designs are read caches for speeding up reads from memory. In some cases, write caches are used for speeding up writes to memory. However in the case of writes to system memory (e.g. disks), data is still being written to system memory directly every time they occur, while being written into cache at the same time. This is done because of concern for loss of updated data files in case of power loss. If the write data is only stored in the cache memory (volatile) a loss of power will result in the new updated files being lost from cache before having the old data updated in system memory (non-volatile). The system will then be operating on the old data when

these files are used in further processing. The need to write to main memory every time defeats the caching mechanism for writes. Read caching does not have this concern since the data that could be lost from cache has a backup on disk.

In the present invention, a system of Flash EEprom is used to provide non-volatile memory in place of traditional system memories such as disk storage. However, Flash EEprom memory is subject to wearing out by excessive program/erase cycles. Even with the improved Flash EEprom memory device as disclosed in copending U.S. patent applications, Serial No. 204,175 and one entitled "Multi-state EEprom Read and Write Circuits and Techniques," by Sanjay Mehrotra and Dr. Eliyahou Harari filed on the same day as the present application, the endurance limit is approximately 10^5 program/erase cycles. In a ten-year projected life time of the device, this translates to a limit of one program/erase cycle per 5 minutes. This may be marginal in normal computer usage.

To overcome this problem, a cache memory is used in a novel way to insulate the Flash EEprom memory device from enduring too many program/erase cycles. The primary function of the cache is to act on writes to the Flash EEprom memory and not on reads of the Flash EEprom memory, unlike the case with traditional caches. Instead of writing to the Flash EEprom memory every time the data is updated, the data may be operated on several times in the cache before being committed to the Flash EEprom memory. This reduces the number of writes to the Flash EEprom memory. Also, by writing mostly into the faster cache memory and reducing the number of writes to the slower Flash EEprom, an additional benefit is the increase in system write throughput.

A relatively small size cache memory is quite effective to implement the present invention. This helps to overcome the problem of data loss in the volatile cache memory during a power loss. In that event, it is relatively easy to have sufficient power reserve to maintain the cache memory long enough and have the data dumped into a non-volatile memory such as a specially reserved space in the Flash EEprom memory. In the event of a power down or and power loss to the system, the write cache system may be isolated from the system and a dedicated rechargeable power supply may be switch in only to power the cache system and the reserved space in the Flash EEprom memory.

Figure 8 illustrates schematically a cache system 701 as part of the controller, according to the present invention. On one hand the cache system 701 is connected to the Flash EEprom memory array 33. On the other hand it is connected to the microprocessor system (not shown) through a host interface 703. The cache system 701 has two memories. One is a cache memory 705 for temporarily holding write data files. The other is a tag memory 709 for storing relevant information about the data files held in the cache memory 705. A memory timing/control circuit 713 controls the writing of data files from the cache memory 705 to the Flash EEprom memory 33. The memory control circuit 713 is responsive to the information stored in the tag memory as well as a power sensing input 715 with is connected through the host interface 703 via a line 717 to the power supply of the microprocessor system. A power loss in the microprocessor system will be sensed by the memory control circuit 713 which will then down load all the data files in the volatile cache memory 705 to the non-volatile Flash EEprom memory 33.

In the present invention, the Flash EEprom memory array 33 is organized into sectors (typically 512 byte size) such that all memory cells within each sector are erasable together. Thus each sector may be considered to store a data file and a write operation on the memory array acts on one or more such files.

During read of a new sector in the Flash EEprom memory 33, the data file is read out and sent directly to the host through the controller. This file is not used to fill the cache memory 705 as is done in the traditional cache systems.

After the host system has processed the data within a file and wishes to write it back to the Flash EEprom memory 33, it accesses the cache system 701 with a write cycle request. The controller then intercepts this request and acts on the cycle.

In one embodiment of the invention, the data file is written to the cache memory 705. At the same time, two other pieces of information about the data file are written to a tag memory 709. The first is a file pointer which identifies the file present in the cache memory 705. The second is a time stamp that tells what time the file was last written into the cache memory. In this way, each time the host wishes to write to the Flash EEprom memory 33, the data file is actually first stored in the cache memory 705 along with pointers and time stamps in the tag memory 709.

In another embodiment of the invention, when a write from the host occurs, the controller first checks to see if that file already existed in the cache memory 705 or has been tagged in the tag memory 709. If it has not been tagged, the file is written to the Flash memory 33, while its identifier and time stamp are written to the tag memory 709. If the file already is present in the cache memory or has been tagged, it is updated in the cache memory and not written to the Flash memory. In this way only infrequently used data files are written into the Flash memory while frequently used data files are trapped in the cache memory.

In yet another embodiment of the invention, when a write from the host occurs, the controller first checks to see if that data file has been last written anywhere within a predetermined period of time (for example, 5 minutes). If it has not, the data file is written to the Flash memory 33, while its identifier and time stamp are written to the tag memory 709. If the data file has been last written within the predetermined period of time, it is written into the cache memory 705 and not written to the Flash memory. At the same time, its identifier and time stamp are written to the tag memory 709 as in the other embodiments. In this way also, only infrequently used data files are written into the Flash memory while frequently used data files are trapped in the cache memory.

In all embodiments, over time the cache memory 705 will start to fill up. When the controller has detected that some predetermined state of fullness has been reached, it begins to archive preferentially some files over others in the cache memory 705 by writing them to the Flash memory 33.

In either embodiments, over time the cache memory 705 will start to fill up. When the controller has detected that some predetermined state of fullness has been reached, it begins to archive preferentially some files over others in the cache memory 705 by writing them to the Flash memory 33. The file identifier tag bits for these files are then reset, indicating that these files may be written over. This makes room for new data files entering the cache memory.

The controller is responsible for first moving the least active files back into the Flash memory 33 to make room for new active files. To keep track of each file's activity level, the time stamp for each file is incremented by the controller at every time step unless reset by a new activity of the file. The timing is provided by timers 711. At every time step (count), the controller systematically accesses each data file in the cache memory and reads the last time stamp written for this data file. The controller then increments the time stamp by another time step (i.e. increments the count by one).

Two things can happen to a file's time stamp, depending on the activity of the file. One possibility is for the time stamp to be reset in the event of a new activity occurring. The other possiblity is that no new activity occurs for the file and the time stamp continues to increment until the file is removed from the cache. In practice a maximum limit may be reached if the time stamp is allowed to increase indefinitely. For example, the system may allow the time stamp to increment to a maximum period of inactivity of 5 minutes. Thus, when a data file is written in the cache memory, the time stamp for the file is set at its initial value. Then the time stamp will start to age, incrementing at every time step unless reset to its initial value again by another write update. After say, 5 minutes of inactivity, the time stamp has incremented to a maximum terminal count.

In one embodiment of keeping count, a bit can be shifted one place in a shift register each time a count increment for a file occurs. If the file is updated (a new activity has occurred) the bit's location will be reset to the initial location of the shift register. On the other hand, if the file remains inactive the bit will eventually be shifted to the terminal shift position. In another embodiment, a count value for each file is stored and incremented at each time step. After each increment, the count value is compared to a master counter, the difference being the time delay in question.

Thus, if a file is active its incremented time stamp is reset back to the initial value each time the data file is rewritten. In this manner, files that are constantly updated will have low time stamp identifiers and will be kept in cache until their activity decreases. After a period of inactivity has expired, they acquire the maximum time stamp identifiers. The inactive files are eventually archived to the Flash memory freeing space in the cache memory for new, more active files. Space is also freed up in the tag memory when these inactive files are moved to the Flash memory.

At any time when room must be made available for new data files coming into the cache memory, the controller removes some of the older files and archives them to the Flash memory 33. Scheduling is done by a memory timing/control circuit 713 in the controller. The decision to archive the file is based on several criteria. The controller looks at the frequency of writes occurring in the system and looks at how full the cache is. If there is still room in the cache, no archiving need to be done. If more room is needed, the files with the earliest time stamps are first removed and archived to the Flash memory.

Although the invention has been described with implementation in hardware in the controller, it is to be understood that other implementations are possible. For example, the cache system may be located elsewhere in the system, or be implemented by software using the existing microprocessor system. Such variations are within the scope of protection for the present invention.

The Profile of how often data is written back to the Flash memory is determined by several factors. It depends on the size of the cache memory and the frequency of writes occurring in the system. With a small cache memory system, only the highest frequency files will be cached. Less frequently accessed files will also be cached with increasing cache memory size. In the present invention, a relatively cheap and small amount of cache mem-

ory, preferably about 1 Mbyte, may be used to good advantage. By not constantly writing the most active files (the top 5%), the write frequency of the Flash EEprom may be reduced from the usual one every millisecond to one every 5 minutes. In this way the wear-out time for the memory can be extended almost indefinitely. This improvement is also accompanied by increased system performance during write.

Incorporating time tag into the write cache concept has the avantage that the size of the write cache buffer memory can be relatively small, since it is used only to store frequently written data files, with all other files written directly into the Flash EEprom memory. A second advantage is that the management of moving data files in and out of the write cache buffer can be automated since it does not require advanced knowledge of which data files are to be called next.

The various aspects of the present invention that have been described co-operate in a system of Flash EEprom memory array to make the Flash EEprom memory a viable alternative to conventional non-volatile mass storage devices.

While the embodiments of the various aspects of the present invention that have been described are the preferred implementation, those skilled in the art will understand that variations thereof may also be possible. Therefore, the invention is entitled to protection within the full scope of the appended claims.

Further, applicant considers as its invention, or inventions, all novel, non-obvious subject matter which is disclosed in this application, and, further, all known equivalents thereto, and expressly reserves the right to submit additional claims of varying scope, which may be narrower or broader than the claims initially filed, drawn to all aspects of the disclosed subject matter and equivalents, even if said subject matter was not set forth in the claims as initially filed, and further reserves the right to amend the claims as initially filed to render them broader or narrower by deleting or adding recitations to said claims as initially filed.

Claims

1. A Flash EEprom system comprising:
one or more integrated circuit chips each having an array of Flash EEprom cells partitioned into a plurality of sectors, each sector addressable for erase such that all cells therein are erasable simultaneously;
means for selecting a plurality of sectors among the one or more chips for erase operation; and
means for simultaneously performing the erase operation on only the plurality of selected sectors.

2. A Flash EEprom system as in claim 1, including
read or write operations on chips which have been enabled by a chip select signal, wherein the erase operation is performed on chips without regard to the chip select signal.

3. A Flash EEprom system as in claim 1, wherein the erase operation may be performed on the plurality of sector selected for erase operation, while read, write or other operations may be performed on any other device not selected for erase operation.

4. The Flash EEprom system according to claim 1, further comprising:
means for individually removing any one or combination of sectors from the plurality of selected sectors, such that said removed sectors are prevented from further erase during the erase operation.

5. The Flash EEprom system according to claim 1, further comprising:
means for simultaneously deselecting all sectors.

6. The Flash EEprom system according to claim 1, wherein the selecting means further comprises:
individual register associated with each sector for holding a status to indicate whether the sector is selected or not.

7. The Flash EEprom system according to claim 6, wherein the simultaneously erasing means is responsive to the status in each of the individual registers, such that only the selected sectors are included in the erasing.

8. The Flash EEprom system according to claim 6, wherein any one or combination of the individual registers indicating a selected status are individually resettable to an un-selected status.

9. The Flash EEprom system according to claim 6, wherein all the individual registers are simultaneously resettable to a status indicating the associated sectors as not selected.

10. A system for correcting errors from defective cells within an array of Flash EEprom cells, comprising:
substitute cells;
means for substituting one or more of the defective cells with a corresponding number of substitute cells.

11. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 10 wherein the substituting means also applies automatically to new defective cells as soon as they are detected.

12. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 10, said array being partitioned into a plurality of Flash erasable sectors such that all cells within each sector are erasable at once,

wherein the substitute cells are in the same sector as the defective cells.

13. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 11, further comprising a defect map for storing defect pointers which link the addresses of the defective cells to that of the corresponding substitute cells.

14. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 13, wherein the defect map for said defective cells are located in the same sector as said defective cells.

15. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 10, said array being partitioned into a plurality of Flash erasable sectors such that all cells within each sector are erasable at once, wherein the substitute cells are in the same sector as the defective cells when the number of defective cells in the sector does not exceed a predetermined number, and the substitute cells are in a different sector when the number is exceeded.

16. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 15, wherein said sector is replaced in its entirety by a substitute sector when said number is exceeded.

17. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 15 wherein the substituting means also applies automatically new defective cells as soon as they are detected.

18. A system for correcting errors from defective cells within an array of Flash EEprom cells as in claim 17, including the use of error correction codes.

19. A system for correcting bad data in defective cells within an array of Flash EEprom cells, comprising:
substitute cells for storing good data intended for the defective cells;
means for substituting the bad data in one or more of the defective cells with the good data in the corresponding substitute cells when the defective cells are accessed.

20. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in clain 19, further comprising means for automatically saving the good data intended to be written to the defective cells to the corresponding substitute cells, thereby perserving the integrity of the good data.

21. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 20 wherein the substituting means also applies automatically to new defective cells as soon as they are detected.

22. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 20, said array being partitioned into a plurality of Flash erasable sectors such that all cells within each sector are erasable at once, and data is stored therein, wherein the substituting means applies after the data including the bad data has been accessed.

23. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 20, said array being partitioned into a plurality of Flash erasable sectors such that all cells within each sector are erasable at once, wherein the substitute cells are in the same sector as the defective cells.

24. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 20, further comprising a defect map for storing defect pointers which link the addresses of the defective cells to that of the corresponding substitute cells.

25. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 24, wherein the defect map for said defective cells are located in the same sector as said defective cells.

26. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 19, said array being partitioned into a plurality of Flash erasable sectors such that all cells within each sector are erasable at once, wherein the substitute cells are in the same sector as the defective cells when the number of defective cells in the sector does not exceed a predetermined number, and the substitute cells are in a different sector when the number is exceeded.

27. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 26, wherein said sector is replaced in its entirety by a substitute sector when said number is exceeded.

28. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 26, wherein the substituting means also applies automatically to newly occurring defective cells.

29. A system for correcting bad data in defective cells within an array of Flash EEprom cells as in claim 28, including use of error correction codes.

30. An improved system for writing data files into a Flash EEprom memory comprising:
a cache memory for temporarily storing data files intended for the Flash EEprom memory, said cache memory able to undergo significantly more write/erase cycles than the Flash EEprom memory;
means responsive to a system write to the Flash EEprom memory for writing data files into the cache memory instead of the Flash EEprom mem-

ory;
means for identifying each data file in the cache memory;
means for determining the time since each data file was last written; and
means for first moving data file having the longest time since last written from the cache memory to the Flash EEprom memory when additional space for new data files is required in the cache memory, thereby substantially reducing the number of actual writes and associated stress to the Flash EEprom memory.

31. The improved system as in claim 30, further comprising:
a backup non-volatile memory for downloading the data files in the cache memory thereto; and
means responsive to an impending power loss for down loading the data files in the cache memory to the backup memory, thereby saving the data files from the possibly volatile cache memory.

32. The improved system as in claim 30, wherein the backup memory is part of the Flash EEprom memory.

33. The improved system as in claim 30, wherein the cache memory has a significantly faster access time than that of the Flash EEprom memory.

34. The improved system as in claim 30, including a controller circuit chip for controlling the operations of the Flash EEprom memory, wherein the improved system is part of the controller circuit chip.

35. The improved system as in claim 30, including a microprocessor system and random access memory, wherein the improved system is implemented by software in the microprocessor system with random access memory.

36. An improved system for writing data files into a Flash EEprom memory comprising:
a cache memory for temporarily storing data files intended for the Flash EEprom memory, said cache memory able to undergo significantly more write/erase cycles than the Flash EEprom memory;
means responsive to a system write to the Flash EEprom memory for writing data files into the cache memory instead of the Flash EEprom memory;
a tag memory for storing the identity of data files and the time each data file was last written; and
means for first moving data file having the longest time since last written from the cache memory to the Flash EEprom memory when additional space for new data files is required in the cache memory, thereby substantially reducing the number of actual writes and associated stress to the Flash EEprom memory.

37. The improved system as in claim 36, further comprising:
a backup non-volatile memory for downloading the data files in the cache memory thereto; and
means responsive to an impending power loss for down loading the data files in the cache memory to the backup memory, thereby saving the data files from the possibly volatile cache memory.

38. The improved system as in claim 36, wherein the backup memory is part of the Flash EEprom memory.

39. The improved system as in claim 36, wherein the cache memory has a significantly faster access time than that of the Flash EEprom memory.

40. The improved system as in claim 36, including a controller circuit chip for controlling the operations of the Flash EEprom memory, wherein the improved system is part of the controller circuit chip.

41. The improved system as in claim 36, including a microprocessor system and random access memory, wherein the improved system is implemented by software in the microprocessor system with random access memory.

42. An improved system for writing data files into a Flash EEprom memory comprising:
a cache memory for temporarily storing data files intended for the Flash EEprom memory, said cache memory able to undergo significantly more write/erase cycles than the Flash EEprom memory;
means responsive to a system write to the Flash EEprom memory for writing a data file either into the Flash EEprom memory or instead into the cache memory, said responsive means writing to the Flash EEprom when the a previous copy of said data file is not present in the cache memory, and writing to the cache memory when a previous copy of said data file is present in the cache memory; and
means for first moving data files having the longest times since last written from the cache memory to the Flash EEprom memory when additional space for new data files is required in the cache memory, thereby substantially reducing the number of actual writes and associated stress to the Flash EEprom memory.

43. The improved system as in claim 42, further comprising:
a backup non-volatile memory for downloading the data files in the cache memory thereto; and
means responsive to an impending power loss for down loading the data files in the cache memory to the backup memory, thereby saving the data files from the possibly volatile cache memory.

44. The improved system as in claim 42, wherein the backup memory is part of the Flash EEprom memory.

45. The improved system for writing data files into a Flash EEprom memory according to claim

42. wherein said responsive means for writing includes a tag memory for storing the identity of data files and the time each data file was last written, and wherein said responsive means writing to the Flash EEprom when said data file is not tagged in the tag memory, and writing to the cache memory when said data file is tagged in the tag memory.

46. An improved system for writing data files into a Flash EEprom memory comprising:
a cache memory for temporarily storing data files intended for the Flash EEprom memory, said cache memory able to undergo significantly more write/erase cycles than the Flash EEprom memory;
means responsive to a system write to the Flash EEprom memory for writing a data file either into the Flash EEprom memory or instead into the cache memory, said responsive means writing to the Flash EEprom when said data file is last written after the predetermined period of time, and writing to the cache memory when said data file is last written within a predetermined period of time; and
means for first moving data files having the longest times since last written from the cache memory to the Flash EEprom memory when additional space for new data files is required in the cache memory, thereby substantially reducing the number of actual writes and associated stress to the Flash EEprom memory.

47. The improved system as in claim 46, wherein the cache memory has a significantly faster access time than that of the Flash EEprom memory.

48. The improved system as in claim 46, including a controller circuit chip for controlling the operations of the Flash EEprom memory, wherein the improved system is part of the controller circuit chip.

49. The improved system as in claim 46, including a microprocessor system and random access memory, wherein the improved system is implemented by software in the microprocessor system with random access memory.

50. An improved system for writing data files into a Flash EEprom memory comprising:
a cache memory for temporarily storing data files intended for the Flash EEprom memory, said cache memory able to undergo significantly more write/erase cycles than the Flash EEprom memory;
a tag memory for storing the identity of data files and the time each data file was last written;
means responsive to a system write to the Flash EEprom memory for writing a data file either into the Flash EEprom memory or instead into the cache memory, said responsive means writing to the Flash EEprom when the data file is not identified in the tag memory, and writing to the cache memory when the data file is identified in the tag memory; and
means for moving first the data files having the longest times since last written from the cache memory to the Flash EEprom memory when additional space for new data files is required in the cache memory, thereby substantially reducing the number of actual writes and associated stress to the Flash EEprom memory.

51. The improved system as in claim 50, further comprising:
a backup non-volatile memory for downloading the data files in the cache memory thereto; and
means responsive to an impending power loss for down loading the data files in the cache memory to the backup memory, thereby saving the data files from the possibly volatile cache memory.

52. The improved system as in claim 50, wherein the backup memory is part of the Flash EEprom memory.

53. The improved system as in claim 50, wherein the cache memory has a significantly faster access time than that of the Flash EEprom memory.

54. The improved system as in claim 50, including a controller circuit chip for controlling the operations of the Flash EEprom memory, wherein the improved system is part of the controller circuit chip.

55. The improved system as in claim 50, including a microprocessor system and random access memory, wherein the improved system is implemented by software in the microprocessor system with random access memory.

56. A memory card adapted to plug into a computer system in a manner to communicate with a system bus and a standard power supply, comprising the following mounted thereon:
a plurality of EEprom integrated circuit chips, each of said chips including:
a large number of individually addressable storage cells organized into a plurality of sectors, each sector containing a plurality of said storage cells,
a plurality of spare storage cells within any of said sectors,
means responsive to signals on said system bus for erasing all cells in one or more designated sectors without erasing cells in others of said sectors,
means responsive to signals on said system bus for reading the state of addressed storage cells,
means responsive to signals on said system bus for programming addressed storage cells to a predetermined state, and
means responsive to an unsuccessful attempt to either program or erase a storage cell within one of said sectors for substituting one of said spare storage cells therefore while maintaining operation of the remaining cells of said sector.

57. The memory card according to claim 56

which additionally comprises a cache memory mounted on said card, and wherein said programming means includes means for initially programming said cache memory rather than said EEprom memory, said reading means includes means for initially determining whether the cache memory contains data to be read, and said programming means additionally includes means responsive to said cache memory becoming full for writing its oldest unused block of data into said EEprom memory, thereby to make room for new data in said cache memory.

58. The memory card as in claim 56, wherein each of said chips further includes a plurality of spare sectors, and wherein said substituting means also substitutes one of said spare sectors for one of said sectors when a predetermined number of cells in said one of said sector become defective.

59. The memory card as in claim 58, including means for performing error correction using error correction codes.

60. The memory card as in claim 56, including a controller and an interface connected to the system bus, said controller being adapted to be responsive to commands intended for a standard magnetic disk drive storage system connectable to the computer system, thereby emulating said disk drive system.

61. The memory card as in claim 56, in which various operating voltages are required for various operations of the EEprom chips, including means for generating the various operating voltages from the standard power supply.

62. A storage system incorporating therein the memory card of claim 56, comprising:
a controller for controlling the operation of the EEprom chips;
means for generating voltages for the operation of the EEprom chips;
means for error correction in the operation of the storage system; and
means for interfacing the storage system to a computer system.

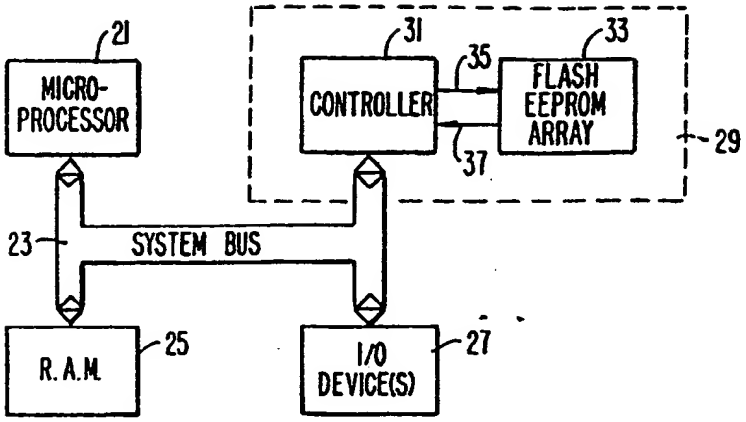


FIG._1A.

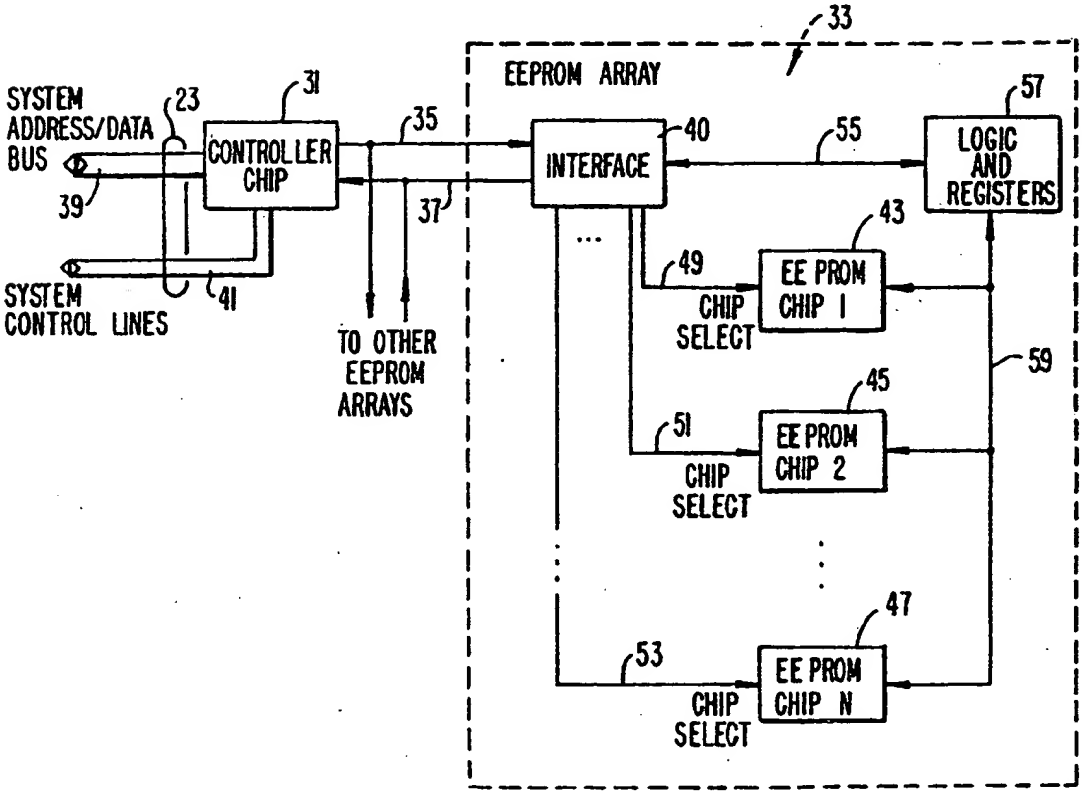


FIG._1B.

FH 008018

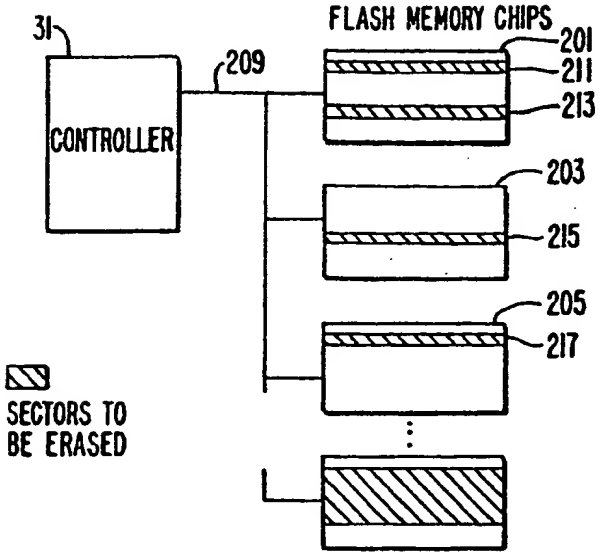


FIG._2.

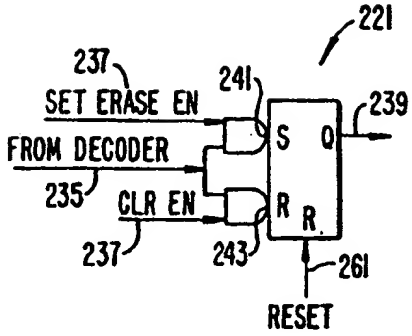


FIG._3B.

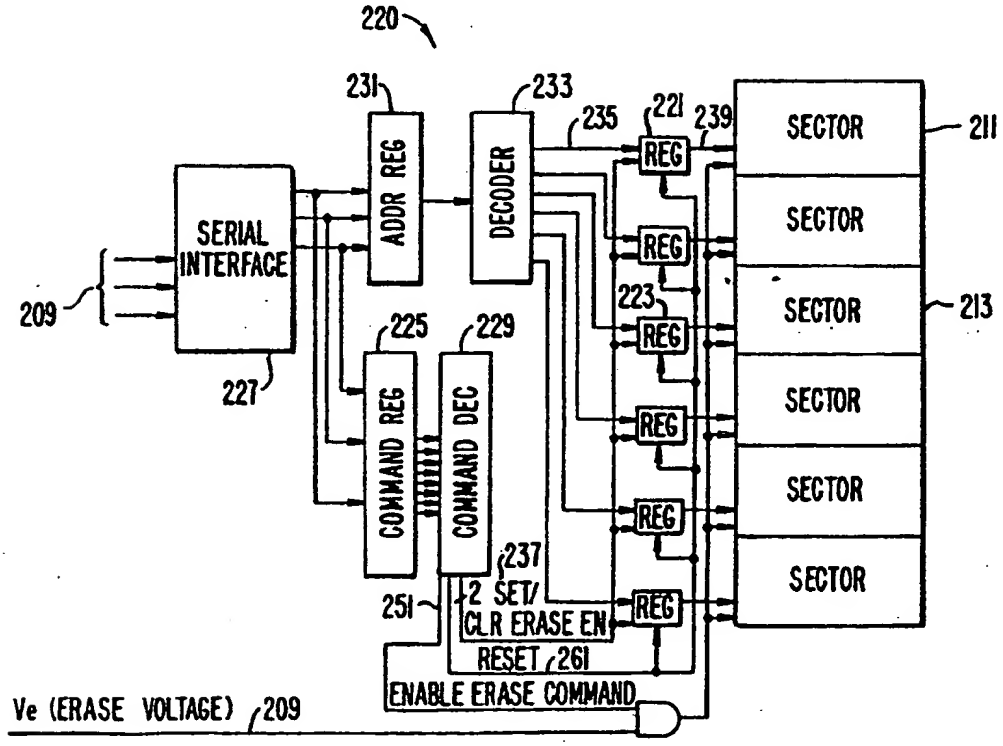


FIG._3A.

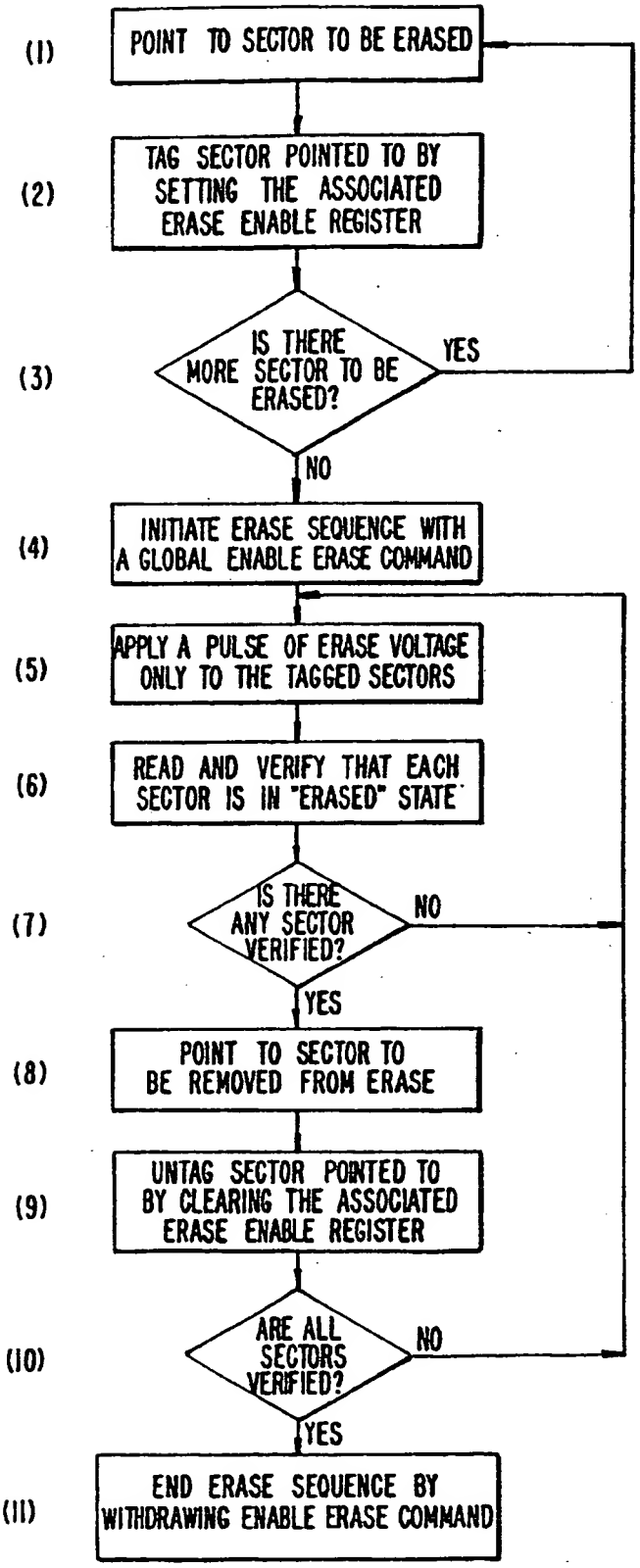
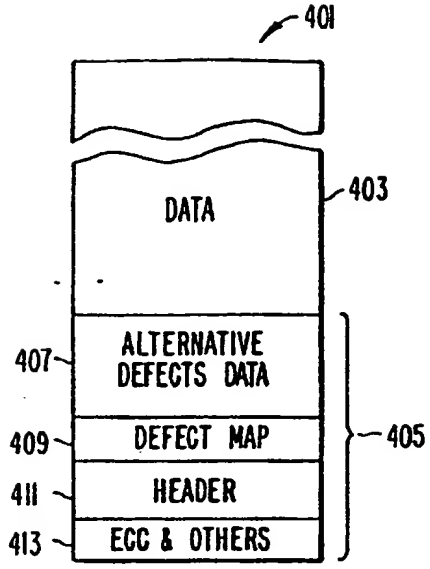

(1)
POINT TO SECTOR TO BE ERASED
(2)
TAG SECTOR POINTED TO BY SETTING THE ASSOCIATED ERASE ENABLE REGISTER
(3)
IS THERE MORE SECTOR TO BE ERASED?
YES
NO
(4)
INITIATE ERASE SEQUENCE WITH A GLOBAL ENABLE ERASE COMMAND
(5)
APPLY A PULSE OF ERASE VOLTAGE ONLY TO THE TAGGED SECTORS
(6)
READ AND VERIFY THAT EACH SECTOR IS IN "ERASED" STATE
(7)
IS THERE ANY SECTOR VERIFIED?
NO
YES
(8)
POINT TO SECTOR TO BE REMOVED FROM ERASE
(9)
UNTAG SECTOR POINTED TO BY CLEARING THE ASSOCIATED ERASE ENABLE REGISTER
(10)
ARE ALL SECTORS VERIFIED?
NO
YES
(11)
END ERASE SEQUENCE BY WITHDRAWING ENABLE ERASE COMMAND

FIG._4.



SECTOR PARTITION

FIG._5.

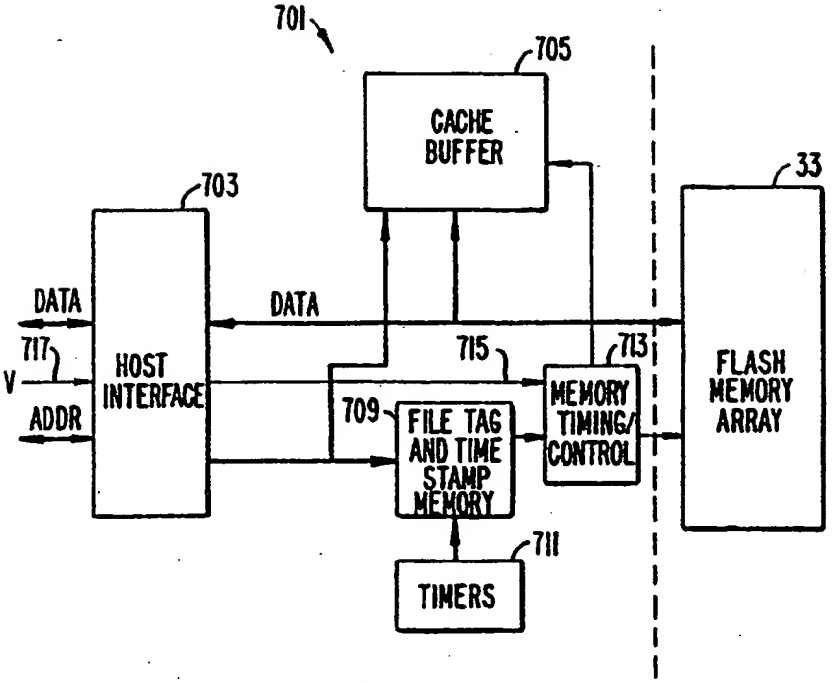
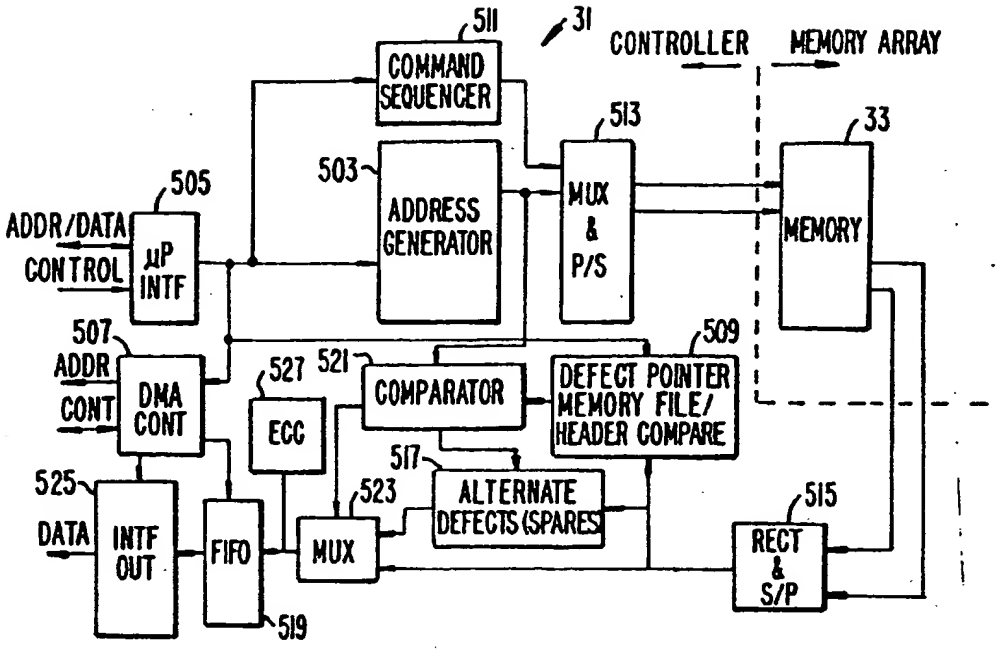
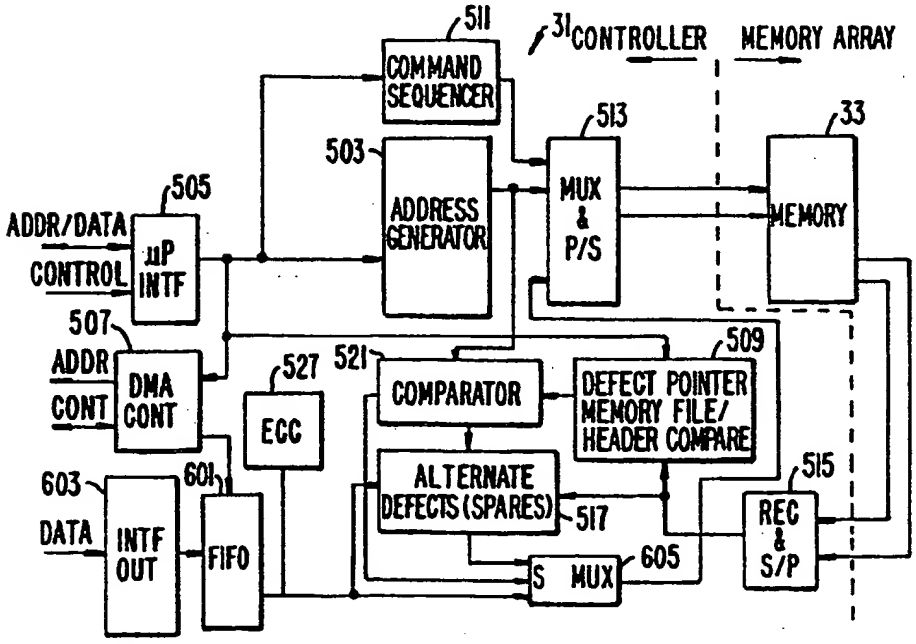


FIG._8.



READ DATA PATH CONTROL

FIG._6.



WRITE DATA PATH CONTROL

FIG._7.

FH 008022

IN THE UNITED STATES DISTRICT COURT FOR THE
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

LEXAR MEDIA, INC., a Delaware corporation,	§	CASE NO. 00-4770 MJJ
	§	
Plaintiff,	§	
	§	
v.	§	
	§	**PLAINTIFF LEXAR**
PRETEC ELECTRONICS CORP.,	§	**MEDIA INC.'S**
a California corporation;	§	**RESPONSE TO**
PNY TECHNOLOGIES, INC.,	§	**DEFENDANT'S PRETEC**
a Delaware corporation; MEMTEK PRODUCTS,	§	**ELECTRONICS CORP.'S**
INC., a California corporation; and C-ONE	§	**FIRST SET OF**
TECHNOLOGY CORPORATION,	§	**REQUESTS FOR**
a Taiwan corporation,	§	**PRODUCTION OF**
	§	**DOCUMENTS**
Defendants	§	
	§	
AND RELATED CROSS-ACTIONS	§	

LEXAR MEDIA, INC.'S OBJECTIONS AND RESPONSES TO PRETEC ELECTRONICS CORP.'S FIRST SET OF REQUESTS FOR PRODUCTION OF DOCUMENTS

PROPOUNDING PARTY:	**Defendant, PRETEC ELECTRONICS CORP.**
RESPONDING PARTY:	**Plaintiff, LEXAR MEDIA, INC.**
SET NO.:	**First**

Plaintiff Lexar Media, Inc. ("Lexar") provides the following responses and objections to defendant Pretec Electronics Corp.'s ("Pretec") first set of requests for production of documents. These responses represent Lexar's reasonable efforts to provide the information requested based upon information in its possession, custody, or control, and based upon its current knowledge and understanding. Lexar reserves the right to alter or amend its responses as set forth herein, and otherwise to assert factual and legal

contentions as additional facts are ascertained and analyses are made. These responses and objections should be treated as confidential pursuant to Local Patent Rule 2-2.

GENERAL OBJECTIONS

1. Lexar reserves all objections and other questions as to competency, relevance, materiality, privilege or admissibility as evidence in any subsequent proceeding in or trial of this or any other action for any purpose whatsoever of its responses herein.
2. Lexar objects to these requests to the extent that they seek to impose obligations on Lexar that are broader than or inconsistent with the Federal Rules of Civil Procedure, applicable Local Rules, or Court orders.
3. Lexar objects to these requests to the extent that they seek information not relevant to any claim or defense of any party in this litigation, not reasonably calculated to lead to the discovery of admissible evidence, or that fall outside the scope of discoverable information under the Federal Rules of Civil Procedure and/or Local Rules of this Court.
4. Lexar objects to these requests to the extent that they seek the disclosure of information protected by the attorney-client privilege, work product immunity, or any other applicable privilege, immunity or protection. Lexar does not intend to produce such privileged or protected documents or information, and the inadvertent or unintentional disclosure of such shall not be deemed a waiver of any such privilege or protection. Lexar expressly reserves the right to object to the introduction at trial or any other use of such information that may be inadvertently or unintentionally disclosed.
5. Lexar objects to these requests to the extent that they are vague, overly broad, unduly burdensome or purport to impose duties or obligations on Lexar beyond the duties or obligations imposed by the Federal Rules of Civil Procedure or fail to describe information sought with the required reasonable particularity.
6. Lexar objects to these requests to the extent that they seek information not in the possession, custody, or control of Lexar.
7. Lexar objects to these requests under the Federal Rules of Civil Procedure to the extent that they seek information that is already known to Pretec or that may be derived or ascertained from documents produced by Lexar or from examination of such documents and for which the burden of deriving or ascertaining the information is

substantially the same for Pretec as it is for Lexar.

8. Lexar objects to these requests to the extent that they lack a limitation on the time period for which responsive information is sought and, therefore, are burdensome, oppressive, overly broad, and not reasonably calculated to lead to the discovery of admissible evidence.
9. Lexar objects to these requests to the extent they are duplicative or redundant of other requests. Each document, thing or response produced in response to one request is deemed to be produced in response to every other request for which it may be produced.
10. Lexar objects to producing any trade secret or confidential information protected by third-party non-disclosure obligations.
11. Lexar's specific objections to any Request are not intended to preclude, override or withdraw any of the foregoing general objections.

SPECIFIC OBJECTIONS TO INSTRUCTIONS

1. Lexar objects to Pretec's instruction to "set forth fully in the objection the facts" upon which Lexar objects to a request that would otherwise require the production of privileged information. Because a given Request could call for the production of numerous privileged documents, and because "the facts upon which [Lexar] rely as the basis for [its] objection" will be specific to each withheld document, this instruction is unduly burdensome.
2. Lexar objects to Pretec's instructions regarding a privilege log to the extent that it requests information in excess of the information required under the Federal Rules or other applicable rules and standards.

SPECIFIC OBJECTIONS TO DEFINITIONS

1. Lexar objects to the definition of "PERSON" provided in paragraph A of Pretec's Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar.
2. Lexar objects to the definition of "YOU," "YOUR," "PLAINTIFF," and "LEXAR" provided in paragraph B of Pretec's Notes and Definitions to the extent it purports

that these definitions include individuals or entities not under the control of Lexar, or to the extent they purport to require Lexar to collect information from individuals or entities not presently under Lexar's direct control, or to the extent they compel Lexar to produce or collect information from individuals or entities whom Lexar has no right or obligation to collect information. Also, Lexar objects to this definition to the extent the definition requires identification or production of documents or things not within the possession, custody, or control of Lexar.

3. Lexar objects to the definition of "DOCUMENT" provided in paragraph R of Pretec's Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar. Also, Lexar objects to this definition to the extent the definition requires disclosure of information protected by the attorney-client privilege, work product immunity, or any other applicable privilege, immunity or protection.
4. Lexar objects to the definition of "THINGS" provided in paragraph S of Pretec's Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar. Also, Lexar objects to this definition to the extent the definition requires disclosure of information protected by the attorney-client privilege, work product immunity, or any other applicable privilege, immunity or protection.
5. Lexar objects to the definition of "COMMUNICATION" provided in paragraph T of Pretec's Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar. Also, Lexar objects to this definition to the extent the definition requires disclosure of information protected by the attorney-client privilege, work product immunity, or any other applicable privilege, immunity or protection.
6. Lexar objects to the definition of "RELATES TO" and "RELATING TO" provided in paragraph U of Pretec's Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar.
7. Lexar objects to the definition of "IDENTIFY" provided in paragraph V of Pretec's

Notes and Definitions to the extent it purports to require identification or production of documents or things not within the possession, custody, or control of Lexar.

SPECIFIC OBJECTIONS AND RESPONSES TO REQUESTS FOR PRODUCTION OF DOCUMENTS

REQUEST NO. 1:

Any documents constituting or reflecting any Foreign Patent Application filed outside the United States of America concerning any invention disclosed under any Patents-At-Issue.

RESPONSE TO REQUEST NO. 1:

Lexar objects to Request No. 1 as overly broad and unduly burdensome to the extent that it calls for documents that are neither relevant nor designed to lead to the discovery of relevant evidence as it asks for "any Foreign Patent Application" related to "any invention disclosed under any Patents-At-Issue." Lexar objects to Request No. 1 as vague, overly broad, unduly burdensome, and ambiguous to the extent that it purports to seek information broadly regarding "any Foreign Patent Applications" and "any invention disclosed under any Patents-At-Issue" without requesting such information with reasonable particularity. Based on its understanding of the phrase "any Foreign Patent Application . . . concerning any invention disclosed under any Patents-At-Issue," Lexar understands Request No. 1 to cover Foreign Patent Applications that are directly related to the Patents-At-Issue. Subject to and without waiving the General Objections or the Specific Objections set forth above, Lexar will produce any non-privileged documents responsive to this Request that are located.

REQUEST NO. 2:

Any documents relating to or reflecting any investigation (including prior art investigation) relating to a Foreign Patent Application concerning any invention disclosed under any Patents-At-Issue.

RESPONSE TO REQUEST NO. 2:

Lexar objects to Request No. 2 to the extent it seeks information protected by the attorney-client privilege, the work product immunity doctrine, or any other privilege, immunity or protection. Lexar objects to Request No. 2 as overly broad and unduly

burdensome to the extent that it calls for documents that are neither relevant nor designed to lead to the discovery of relevant evidence as it asks for "any investigation" relating to "a Foreign Patent Application concerning any invention disclosed under any Patents-At-Issue." Lexar objects to Request No. 2 as vague, overly broad, unduly burdensome, and ambiguous to the extent that it purports to seek information broadly regarding "any investigation" and "any invention disclosed under any Patents-At-Issue" without requesting such information with reasonable particularity. Based on its understanding of the phrase "a Foreign Patent Application concerning any invention disclosed under any Patents-At-Issue," Lexar understands Request No. 2 to cover Foreign Patent Applications that are directly related to the Patents-At-Issue. Subject to and without waiving the General Objections or the Specific Objections set forth above, Lexar will produce any non-privileged documents responsive to this Request that are located.

REQUEST NO. 3:

Any documents relating to any Foreign Patent Applications concerning any invention disclosed under any Patents-At-Issue, including but not limited to any communications, discussions, modifications and file wrappers.

RESPONSE TO REQUEST NO. 3:

Lexar objects to Request No. 3 to the extent it seeks information protected by the attorney-client privilege, the work product immunity doctrine, or any other privilege, immunity or protection. Lexar objects to Request No. 3 as overly broad and unduly burdensome to the extent that it calls for documents that are neither relevant nor designed to lead to the discovery of relevant evidence as it asks for "any documents relating to Foreign Patent Applications concerning ~~related to~~ any invention disclosed under any Patents-At-Issue." Lexar objects to Request No. 3 as vague, overly broad, unduly burdensome, and ambiguous to the extent that it purports to seek information broadly regarding "any documents relating to Foreign Patent Applications concerning ~~and~~ any invention disclosed under any Patents-At-Issue" without requesting such information with reasonable particularity. Based on its understanding of the phrase "any documents relating to any Foreign Patent Application concerning any invention disclosed under any Patents-At-Issue," Lexar understands Request No. 3 to cover Foreign Patent Applications that are directly related to the Patents-At-Issue. Subject to and without waiving the General

Objections or the Specific Objections set forth above, Lexar will produce any non-privileged documents responsive to this Request that are located.

REQUEST NO. 4:

All documents and things the identification of which are requested in PNY's second set of interrogatory requests.

RESPONSE TO REQUEST NO. 4:

Lexar objects to Request No. 4 as objectionable on the same basis that PNY's second set of interrogatory requests served on Lexar is objectionable. Subject to and without waiving the foregoing General Objections or the Specific Objections set forth above, Lexar ~~responds to Request No. 4 as follows:~~ will produce the following applications as follows:

USPN: 5,479,638

1. WO9423369A1: FLASH MEMORY MASS STORAGE ARCHITECTURE
2. WO9423432A1: FLASH MEMORY MASS STORAGE ARCHITECTURE
3. WO9510083A1: FLASH MEMORY WITH REDUCED ERASING AND OVERWRITING
4. EP0722585B1: FLASH MEMORY WITH REDUCED ERASING AND OVERWRITING
5. EP0722585A4: FLASH MEMORY WITH REDUCED ERASING AND OVERWRITING
6. EP0722585A1: FLASH MEMORY WITH REDUCED ERASING AND OVERWRITING
7. EP0693216A4: FLASHMEMORY MASS STORAGE ARCHITECTURE
8. EP0693216A1: FLASHMEMORY MASS STORAGE ARCHITECTURE
9. EP0691008B1: FLASH MEMORY MASS STORAGE ARCHITECTURE
10. EP0691008A4: FLASH MEMORY MASS STORAGE ARCHITECTURE
11. EP0691008A1: FLASH MEMORY MASS STORAGE ARCHITECTURE
12. DE69431795C0: MASSENSPEICHERARCHITEKTUR MIT FLASH-SPEICHER
13. DE69430668T2: FLASH-SPEICHER MIT VERRINGERTER LOESCHUNG UND UEBERSCHREIBUNG
14. DE69430668C0: FLASH-SPEICHER MIT VERRINGERTER LOESCHUNG UND UEBERSCHREIBUNG
15. CA2161345AA: FLASHMEMORY MASS STORAGE ARCHITECTURE
16. CA2161344AA: FLASH MEMORY MASS STORAGE ARCHITECTURE
17. AT0228674E: MASSENSPEICHERARCHITEKTUR MIT FLASH-SPEICHER

USPN: 5,818,781

1. WO9720269A1: AUTOMATIC VOLTAGE DETECTION IN MULTIPLE VOLTAGE APPLICATIONS
2. JP0500892T2
3. EP0861468B1: AUTOMATIC VOLTAGE DETECTION IN MULTIPLE VOLTAGE

APPLICATIONS
4. EP0861468A4: AUTOMATIC VOLTAGE DETECTION IN MULTIPLE VOLTAGE APPLICATIONS
5. EP0861468A1: AUTOMATIC VOLTAGE DETECTION IN MULTIPLE VOLTAGE APPLICATIONS
6. DE69627176C0: AUTOMATISCHE SPANNUNGSDETEKTION IN ANWENDUNGEN MIT MEHREREN SPANNUNGEN
7. CN1202255A: AUTOMATIC VOLTAGE DETECTION IN MULTIPLE VOLTAGE APPLICATIONS

USPN: 5,907,856; 5,930,815; 6,145,051; 6,202,138, 6397,314

1. WO9944113C2: INCREASING THE MEMORY PERFORMANCE OF FLASH MEMORY DEVICES BY WRITING SECTORS SIMULTANEOUSLY TO MULTIPLE FLASH MEMORY DEVICES
2. WO9944113A3: INCREASING THE MEMORY PERFORMANCE OF FLASH MEMORY DEVICES BY WRITING SECTORS SIMULTANEOUSLY TO MULTIPLE FLASH MEMORY DEVICES
3. WO9944113A2: INCREASING THE MEMORY PERFORMANCE OF FLASH MEMORY DEVICES BY WRITING SECTORS SIMULTANEOUSLY TO MULTIPLE FLASH MEMORY DEVICES
4. WO9927453A1: ALIGNMENT OF CLUSTER ADDRESS TO BLOCK ADDRESSES WITHIN A SEMICONDUCTOR NON-VOLATILE MASS STORAGE MEMORY
5. WO9918509C2: MOVING SEQUENTIAL SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE
6. WO9918509A1: MOVING SEQUENTIAL SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE
7. WO9844420A1: MOVING SECTORS WITHIN A BLOCK IN A FLASH MEMORY
8. WO9710604A1: METHOD OF AND ARCHITECTURE FOR CONTROLLING SYSTEM DATA WITH AUTOMATIC WEARLEVELING IN A SEMICONDUCTOR NON-VOLATILE MASS STORAGE MEMORY
9. WO0077791A1: METHOD AND APPARATUS FOR DECREASING BLOCK WRITE OPERATION TIMES PERFORMED ON NONVOLATILE MEMORY
10. WO0060605A1: SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY
11. WO0002126A1: METHOD AND APPARATUS FOR PERFORMING ERASE OPERATIONS TRANSPARENT TO A SOLID STATE STORAGE SYSTEM
12. JP2002508862T2
13. JP2000510634T2
14. EP1228510A1: SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY
15. EP1036364A1: ALIGNMENT OF CLUSTER ADDRESS TO BLOCK ADDRESSES WITHIN A SEMICONDUCTOR NON-VOLATILE MASS STORAGE MEMORY
16. EP1029278A1: MOVING SEQUENTIAL SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE

17. EP0980551A1: MOVING SECTORS WITHIN A BLOCK IN A FLASH MEMORY
18. AU7071796A1: METHOD OF AND ARCHITECTURE FOR CONTROLLING SYSTEM DATA WITH AUTOMATIC WEAR LEVELING IN A SEMICONDUCTOR NON-VOLATILE MASS STORAGE MEMORY
19. AU6873898A1: MOVING SECTORS WITHIN A BLOCK IN A FLASH MEMORY
20. AU4862899A1: METHOD AND APPARATUS FOR PERFORMING ERASE OPERATIONS TRANSPARENT TO A SOLID STATE STORAGE SYSTEM
21. AU1538899A1: ALIGNMENT OF CLUSTER ADDRESS TO BLOCK ADDRESSES WITHIN A SEMICONDUCTOR NON-VOLATILE MASS STORAGE MEMORY
22. AU0054741A5: METHOD AND APPARATUS FOR DECREASING BLOCK WRITE OPERATION TIMES PERFORMED ON NONVOLATILE MEMORY
23. AU0040617A5: SPACE MANAGEMENT FOR MANAGING HIGH CAPACITY NONVOLATILE MEMORY
24. AU1517799A1: MOVING SEQUENTIAL SECTORS WITHIN A BLOCK OF INFORMATION IN A FLASH MEMORY MASS STORAGE ARCHITECTURE

REQUEST NO. 5:

All documents and things relating to any investigation you have undertaken, prior to filing the Complaints, to conclude that any of PRETEC's products or devices infringe the Patents-At-Issue.

RESPONSE TO REQUEST NO. 5:

Lexar objects to Request No. 5 ~~to the extent~~ because it seeks information protected by the attorney-client privilege, the work product immunity doctrine, or any other privilege, immunity or protection. Lexar objects to Request No. 5 as vague, overly broad, unduly burdensome, and ambiguous to the extent that it purports to seek information broadly regarding "any investigation" without requesting such information with reasonable particularity. ~~Subject to and without waiving the General Objections or the Specific Objections set forth above, Lexar will produce any non-privileged documents responsive to this Request that are located.~~

Dated: August 4, 2003

WEIL, GOTSHAL & MANGES, L.L.P.

By: _____

David J. Healey

Attorneys for Plaintiff
LEXAR MEDIA, INC.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer: **0 494 329 A1**

(12) EUROPÄISCHE PATENTANMELDUNG

(21) Anmeldenummer: **91100274.9**

(51) Int. Cl.5: **G06F 12/02, G06F 12/06**

(22) Anmeldetag: **10.01.91**

(43) Veröffentlichungstag der Anmeldung:
15.07.92 Patentblatt 92/29

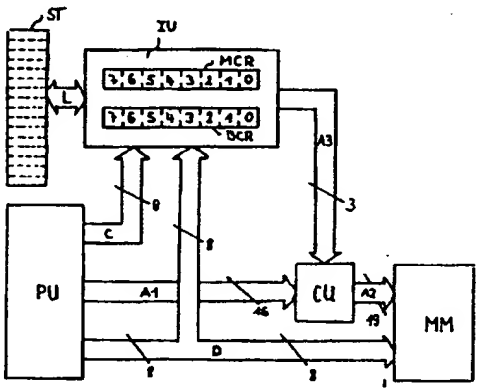
(84) Benannte Vertragsstaaten:
AT BE CH DE DK ES FR GB GR IT LI LU NL SE

(71) Anmelder: **SIEMENS AKTIENGESELLSCHAFT**
Wittelsbacherplatz 2
W-8000 München 2(DE)

(72) Erfinder: **Möller, Udo, Dipl.-Ing.**
Schweizer Strasse 8
W-4100 Duisburg(DE)

(54) Schaltungsanordnung zur Abbildung des logischen Adressraums einer Prozessoreinheit auf den physikalischen Adressraum eines Speichers.

(57) Schaltungsanordnung zur Abbildung des logischen Adreßraums einer Prozessoreinheit (PU) auf den physikalischen Adreßraum eines Speichers (MM) mit einer über Daten- und Steuerleitungen (D,C) mit der Prozessoreinheit (PU) verbundenen Interpretationseinheit (IU), die ein in einen ersten Registerbereich und in einen, durch die Prozessoreinheit zu beschreibenden, zweiten Registerbereich aufgeteiltes Register (BCR) enthält, wobei die Interpretationseinheit (IU) den logischen Zustand der Prozessoreinheit (PU) fortlaufend auswertet, bei bestimmten logischen Zuständen den Inhalt des zweiten Registerbereichs in den ersten Registerbereich einschreibt und den Inhalt des ersten Registerbereichs als Adresse ausgibt, und mit einer über Adreßleitungen (A1,A2,A3) mit der Prozessoreinheit (PU), der Interpretationseinheit (IU) und dem Speicher (MM) verbundenen Verknüpfungseinheit, die aus von der Interpretationseinheit (IU) und der Prozessoreinheit (PU) übertragenen Adressen eine Gesamtadresse für den Speicher (MM) bildet.



EP 0 494 329 A1

Die Erfindung betrifft eine Schaltungsanordnung zur Abbildung des logischen Adreßraums einer Prozessoreinheit auf den physikalischen Adreßraum eines Speichers.

Für Prozessoreinheiten, beispielsweise Mikroprozessoren, Mikrocomputer oder Mikrocontroller, erschließen sich ständig neue Anwendungsgebiete. Damit steigen aber auch die Anforderungen an die Prozessoreinheiten und deren Architektur. Bei einer Vielzahl von Architekturen steht dabei dem logischen Adreßraum, also der Anzahl der durch die Prozessoreinheit darstellbaren Adressen, meist ein um ein Vielfaches größerer physikalischer Adreßraum, das heißt, die Anzahl aller bei einer zur Verfügung stehenden Speicherkapazität möglichen Adressen, gegenüber.

Aufgabe der Erfindung ist es daher, eine Schaltungsanordnung anzugeben, die den logischen Adreßraum einer Prozessoreinheit auf den physikalischen Adreßraum eines Speichers abbildet.

Die Aufgabe wird durch eine Schaltungsanordnung gemaß Patentanspruch 1 gelöst. Ausgestaltungen und Weiterbildungen des Erfindungsgedankens sind Gegenstand von Unteransprüchen.

Die Erfindung wird nachfolgend anhand des in der einzigen Figur der Zeichnung dargestellten Ausführungsbeispiels näher erläutert.

Bei diesem Ausführungsbeispiel wird von einer gegebenen Prozessoreinheit PU und einem gegebenen Speicher MM ausgegangen. Die Prozessoreinheit PU ist an acht Datenleitungen D, an sechszehn erste Adreßleitungen A1 und an acht Steuerleitungen C angeschlossen. Die Datenleitungen D sind zudem mit dem Speicher MM, der darüber hinaus an neunzehn zweite Adreßleitungen A2 angeschlossen ist, und mit einer Interpretationseinheit IU verbunden. Die Interpretationseinheit IU, die ein Bereichskontrollregister BCR und ein Moduskontrollregister MCR enthält, ist zudem mit den Kontrolleitungen C sowie drei dritten Adreßleitungen A3 verschaltet. Außerdem ist ein Stapelregister ST über Verbindungsleitungen L mit der Interpretationseinheit IU gekoppelt. Schließlich ist eine Verknüpfungseinheit CU vorgesehen, die mit ersten, zweiten und dritten Adreßleitungen A1,A2,A3 verbunden ist.

Das Bereichskontrollregister BCR weist eine Breite von 8 Bit auf, wobei drei der vier höherwertigen Bit - Bit 4,5 und 6 - einen ersten Registerbereich und die drei niederwertigen Bit - Bit 0,1 und 2 - einem zweiten Registerbereich zugeordnet sind. Die Prozessoreinheit PU kann auf die beiden Registerbereiche zugreifen, und zwar dergestalt, daß beide Bereiche gelesen werden können, jedoch nur der zweite Bereich beschrieben werden kann. Das Schreiben bzw. Lesen des Bereichskontrollregisters BCR durch die Prozessoreinheit PU erfolgt über die Datenleitungen D. Darüber hinaus kann durch die Interpretationseinheit IU zum einen der Inhalt des ersten Registerbereiches an die Verknüpfungseinheit CU ausgelesen werden und zum anderen der Inhalt des zweiten Registerbereichs in den ersten Registerbereich übernommen werden. Folglich werden zur Übertragung des Inhalts des ersten Registerbereichs an die Verknüpfungseinheit CU lediglich drei dritte Adreßleitungen A3 benötigt. Die verbleibenden Bit des Bereichskontrollregisters BCR - Bit 3 und 7 - können gegebenenfalls für andere Zwecke verwendet werden oder mit einem festen logischen Pegel beaufschlagt werden. Die Verknüpfungseinheit CU bildet aus den sechszehn Bit der von der Prozessoreinheit PU gelieferten Adresse und den drei Bit der von der Interpretationseinheit IU erhaltenen Adresse eine neunzehn Bit breite Gesamtadresse, die über die zweiten Adreßleitungen A2 dem Speicher MM zugeführt wird. Die beiden Adressen werden beispielsweise derart verknüpft, daß die 3 Bit der Interpretationseinheit IU die drei höherwertigen Bit und die 16 Bit der Prozessoreinheit PU die sechszehn niederwertigen Bit der Gesamtadresse ergeben. Somit ist der durch den Speicher MM festgelegte physikalische Adreßraum um drei Bit größer bzw. achtmal so groß als der durch die Prozessoreinheit PU bestimmte logische Adreßraum. Dies kann auch so interpretiert werden, daß der Speicher MM in acht Speicherbereiche eingeteilt ist, die jeweils für sich durch die Prozessoreinheit PU selbst adressierbar sind. Dagegen wird der jeweilige Speicherbereich durch die drei von der Interpretationseinheit IU der Verknüpfungseinheit CU zugeführten Bit ausgewählt. Die im Bereichskontrollregister eingeschriebenen Daten stellen somit Bereichskennzahlen dar, wobei die im ersten Registerbereich stehende Bereichskennzahl den aktuellen Speicherbereich kennzeichnet und der Inhalt des zweiten Registerbereichs den als nächsten zu bearbeitenden Speicherbereich kennzeichnet.

Die Funktionsweise beruht nun darauf, daß die Interpretationseinheit IU den logischen Zustand der Prozessoreinheit PU anhand der auf den Steuerleitungen C übertragenen Signale und gegebenenfalls auch anhand der auf den Datenleitungen D empfangenen Signale ermittelt und bei bestimmten Zuständen entsprechend reagiert. Bei Auftreten bestimmter Zustände wird der Inhalt des zweiten Registerbereichs in den ersten Registerbereich übernommen und damit ein anderer Speicherbereich ausgewählt. Das bedeutet, daß das Beschreiben des Bereichskontrollregisters BCR durch die Prozessoreinheit PU niemals zur gleichen Zeit erfolgt wie eine Anderung des Speicherbereichs, wodurch sich eine Verzögerung zwischen Beschreiben und Freigabe des entsprechenden Speicherbereichs ergibt.

Ein Rücksetzen der Prozessoreinheit PU wird

von der Interpretationseinheit IU erkannt, die daraufhin das Bereichskontrollregister BCR insgesamt gleich Null setzt. Das bedeutet, daß auch der erste und zweite Registerbereich und damit die aktuelle Bereichskennzahl und die nächste Bereichskennzahl gleich Null sind. Die Prozessoreinheit PU beginnt daraufhin mit der Abarbeitung ihres Programms in dem Speicherbereich mit der Bereichskennzahl Null. Während des Programmablaufs wird nun an beliebiger Stelle die Bereichskennzahl des als nächsten ausgewählten Speicherbereichs in den zweiten Registerbereich eingeschrieben. Im weiteren Verlauf kann dann durch bestimmte Steuersignale ein Speicherbereichswechsel durchgeführt werden, indem der Inhalt des zweiten Registerbereichs in den ersten Registerbereich übernommen wird.

In Weiterbildung der Erfindung weist die Interpretationseinheit IU das Moduskontrollregister MCR auf, bei dem ein Bit, beispielsweise das höherwertigste Bit 7, zur Kennzeichnung eines Datenspeichermodus vorgesehen ist. Bei diesem Betriebsfall ist Bit 7 gleich einer logischen Eins. Ein weiteres Bit des Moduskontrollregisters MCR, beispielsweise das Bit 3, signalisiert durch eine logische Null, ob der Stapelspeicher ST gefüllt ist. Darüber hinaus bilden Bit 4,5 und 6 einen dritten Registerbereich und Bit 0,1 und 2 einen vierten Registerbereich.

Bei gesetztem Bit 7, im folgenden Memory-Mode-Bit genannt, des Moduskontrollregisters MCR, erscheint bei Datenzugriffen der Prozessoreinheit PU anstelle des Inhalts des ersten Registerbereichs der Inhalt des dritten Registerbereichs auf den dritten Adreßleitungen A3. Ist das Memory-Mode Bit beispielsweise gleich einer logischen Eins, ändert sich die Adresse auf den dritten Adreßleitungen bei Datenzugriffen nicht. Außerdem kann die Interpretationseinheit IU gegebenenfalls den Inhalt des ersten oder zweiten Registerbereichs in den dritten Registerbereich übertragen.

Zur einfacheren Handhabung von Unterprogrammaufrufbefehlen und Interrupts ist in Weiterbildung der Erfindung der Stapelspeicher ST mit sechszehn jeweils sechs Bit breiten Einzelregistern vorgesehen. Bei bestimmten Unterprogrammaufrufbefehlen werden der erste und zweite Registerbereich des Bereichskontrollregisters in das Stapelregister ST übertragen und ein Speichererweiterungszeiger wird inkrementiert. Daraufhin wird der Inhalt des zweiten Registerbereichs in den ersten Registerbereich übertragen. Beim Verlassen des Unterprogramms mit bestimmten Befehlen wird der Zeiger dekrementiert und die alten Inhalte des ersten und zweiten Registerbereichs werden aus dem Stapelspeicher ausgelesen. Dies dient dazu, um bei eventuell auftretenden Interrupts den bisherigen Zustand zu sichern. Weitere Vereinfachung in der Handhabung von Interrupts werden dadurch erzielt, daß bei Auftreten eines Interrupts der Inhalt des vierten Registerbereiches beim Moduskontrollregister MCR in den ersten und zweiten Registerbereich beim Bereichskontrollregister BCR übertragen werden, nachdem deren Inhalte dem Stapelregister ST übergeben worden sind. Dadurch wird erreicht, daß beim Programmieren Interrupt-Service-Routinen in einem speziellen Speicherbereich abgelegt werden können. Nach einem Reset werden das Memory-Mode-Bit, der dritte Registerbereich und der vierte Registerbereich gleich Null gesetzt.

Schließlich dient vorteilhafterweise ein Stapelregisterüberlaufbit, nämlich Bit 3 des Moduskontrollregisters MCR, dazu, bei weitläufigen Unterprogrammverzweigungen einen bevorstehenden Überlauf des Stapelregisters ST anzuzeigen, um einen Verlust der Programmsteuerung zu vermeiden.

Entscheidend ist, daß eine Freigabe des ausgewählten Speicherbereiches nicht sofort mit dem Beschreiben des entsprechenden Registerbereiches, nämlich des ersten Registerbereiches, stattfindet, sondern nur in Abhängigkeit von bestimmten Befehlen und von diesen wiederum abgeleiteten logischen Zuständen oder bestimmter anderer logischer Zustände der Prozessoreinheit PU. . Bei Sprungbefehlen und Unterprogrammaufrufbefehlen gibt man beispielsweise die Zieladresse an und aktiviert damit gleichzeitig auch einen neuen Speicherbereich. Die Aktivierung nimmt die Interpretationseinheit IU vor, weil diese einen entsprechenden logischen Zustand der Prozessoreinheit PU erkennt. Durch das Stapelregister ST wird nun ermöglicht, daß Unterprogrammaufrufbefehle im gesamten physikalischen Adreßraum möglich sind, da im Stapelspeicher ST die jeweiligen Bereichskennzahlen abgespeichert werden. Ein in der Prozessoreinheit PU vorhandener Stapelspeicher ist nämlich nicht in der Lage, eine erweiterte Adresse abzuspeichern. Außerdem ist der Stapelspeicher ST auch für die Interrupt-Behandlung vorteilhaft. Damit Daten und Befehle nicht im gleichen Speicherbereich liegen müssen, kann bei entsprechenden Befehlen der Datenzugriff auf einen anderen Speicherbereich umgelenkt werden. Die Software-Entwicklung kann vorteilhafterweise weitgehend mit existierenden Tools erfolgen.

Neben dem geringen Schaltungsaufwands zeichnet sich eine erfindungsgemäße Schaltungsanordnung durch Erleichterungen bei der Programmentwicklung aus, da ein Wechsel der Speicherbereiche unabhängig vom Beschreiben des zugeordneten Registers ist, da Interrupt-Service-Routinen nicht in jedem Speicherbereich abgelegt werden müssen, da Daten und Programm nicht im gleichen Speicherbereich abgelegt werden müssen und da Programmverschachtelungen ohne weiteres

möglich sind.

Patentansprüche

1. Schaltungsanordnung zur Abbildung des logischen Adreßraums einer Prozessoreinheit (PU) auf den physikalischen Adreßraum eines Speichers (MM) mit einer über Daten- und Steuerleitungen (D,C) mit der Prozessoreinheit (PU) verbundenen Interpretationseinheit (IU), die ein in einen ersten Registerbereich und in einen, durch die Prozessoreinheit (PU) zu beschreibenden, zweiten Registerbereich aufgeteiltes Register (BCR) enthält, wobei die Interpretationseinheit (IU) den logischen Zustand der Prozessoreinheit (PU) fortlaufend auswertet, bei bestimmten logischen Zuständen den Inhalt des zweiten Registerbereichs in den ersten Registerbereich einschreibt und den Inhalt des ersten Registerbereichs als Adresse ausgibt, und mit einer über Adreßleitungen (A1,A2,A3) mit der Prozessoreinheit (PU), der Interpretationseinheit (IU) und dem Speicher (MM) verbundenen Verknüpfungseinheit, die aus von der Interpretationseinheit (IU) und der Prozessoreinheit (PU) übertragenen Adressen eine Gesamtadresse für den Speicher (MM) bildet.

2. Schaltungsanordnung nach Anspruch 1, **dadurch gekennzeichnet,** daß das Register (BCR) der Interpretationseinheit (IU) durch die Prozessoreinheit (PU) lesbar ist.

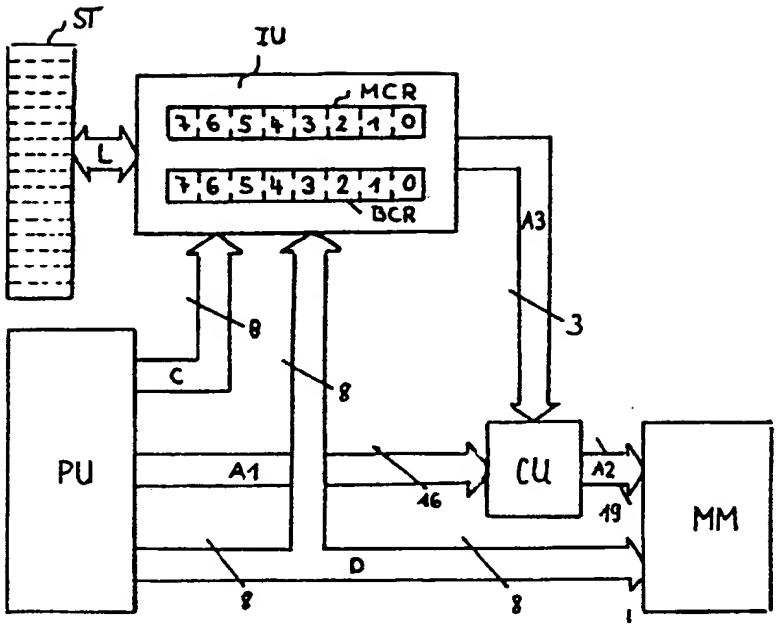
3. Schaltungsanordnung nach Anspruch 1 oder 2, **dadurch gekennzeichnet,** daß ein Stapelregister (ST), an die Interpretationseinheit (IU) angeschlossen ist, in das bei bestimmten logischen Zuständen der Prozessoreinheit (PU) der Inhalt von erstem und zweitem Registerbereich eingelesen wird und das bei anderen bestimmten logischen Zuständen an dem ersten und zweiten Registerbereich ausgelesen wird.

4. Schaltungsanordnung nach einem der Ansprüche 1 bis 3, **dadurch gekennzeichnet,** daß die Interpretationseinheit (IU) ein weiteres Register (MCR) mit zwei weiteren Registerbereichen enthält, in die bei Auftreten bestimmter logischer Zustände jeweils durch die Prozessoreinheit (PU) eingelesen wird.

5. Schaltungsanordnung nach einem der Ansprüche 1 bis 4, **dadurch gekennzeichnet,** daß bei dem weiteren Register (MCR) ein durch die Prozessoreinheit (PU) setzbares Bit vorgesehen ist, von dessen Zustand abhängig bei bestimmten logischen Zuständen der Prozessoreinheit (PU) entweder der erste Registerbereich oder ein weiterer Registerbereich durch die Interpretationseinheit (IU) ausgelesen wird.

6. Schaltungsanordnung nach einem der Ansprüche 1 bis 5, **dadurch gekennzeichnet,** daß bei dem weiteren Register (MCR) ein weiteres Bit vorgesehen ist, das einen Uberlauf des Stapelregisters (ST) der Prozessoreinheit (PU) ankündigt.

7. Schaltungsanordnung nach einem der Ansprüche 1 bis 6, **dadurch gekennzeichnet,** daß die Interpretationseinheit (IU) beim Rücksetzen der Prozessoreinheit (PU) den ersten und zweiten Registerbereich in einen festgelegten Zustand bringt.


ST
IU
MCR
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
DCR
L
A3
3
8
C
8
PU
A1
46
CU
A2
19
MM
D
8
8
1